Sample Chapter

Hello World!

Computer Programming for Kids and Other Beginners

Warren and Carter Sande

MANNING

Hello World!

Computer Programming for Kids and Other Beginners

by Warren Sande
and Carter Sande

Chapter 6

brief contents

Preface xiii
Acknowledgments xix
About this book xxi

1 Getting Started 1

2 Remember This—Memory and Variables 14

3 Basic Math 26

4 Types of Data 38

5 Input 44

6 GUIs—Graphical User Interfaces 52

7 Decisions, Decisions 62

8 Loop the Loop 74

9 Just for You—Comments 89

10 Game Time 94

11 Nested and Variable Loops 99

12 Collecting Things Together—Lists 112

13 Functions 131

14 Objects 146

15 Modules 164

16 Graphics 174

17 Sprites and Collision Detection 202

18 A New Kind of Input—Events 217

19 Sound 239

20 More GUIs 254

21 Print Formatting and Strings 273

22 File Input and Output 290

23 Take a Chance—Randomness 313

24 Computer Simulations 336

25 What's Next? 358

Appendix A Variable Naming Rules 363

Answers to Self-Test Questions 365

Index 393

C H A P T E R 6

GUIs—Graphical User Interfaces

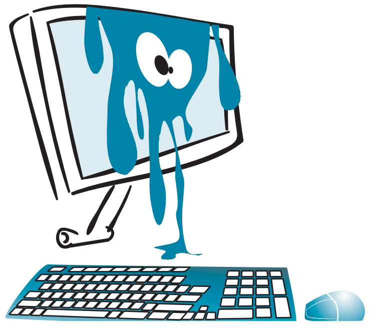
Up until now, all our input and output has been simple text in the IDLE window. But modern computers and programs use lots of graphics. It would be nice if we could have some graphics in our programs. In this chapter, we'll start making some simple GUIs. That means our programs will start to look more like the ones you're used to—with windows, buttons, and so on.

What's a GUI?

GUI is an abbreviation for *graphical user interface*. In a GUI, instead of just typing text and getting text back, the user sees graphical things like windows, buttons, text boxes, etc., and she can use the mouse to click things as well as type on the keyboard. The types of programs we have done so far are *command-line* or *text-mode* programs. A GUI is just a different way of interacting with a program. Programs that have a GUI still have the three basic elements: input, processing, and output. It's just that their input and output are a bit fancier.

TIP

By the way, the acronym GUI is usually pronounced "gooey," instead of saying the letters, like "Gee You Eye." It's okay to have a GUI on your computer, but you should avoid getting anything gooey on your computer. It gets stuck in the keys and makes it hard to type!



Our first GUI

We have already been using a GUI—in fact, several of them. A web browser is a GUI. IDLE is a GUI. Now we're going to make our own GUI. To do this, we're going to get some help from something called EasyGui.

EasyGui is a Python module that makes it very easy to make simple GUIs. We haven't really talked about modules yet (we will in chapter 15), but a module is a way of adding something to Python that isn't already built in.

If you installed Python using the book's installer, you already have EasyGui installed. If not, you can download it from http://easygui.sourceforge.net/.

Installing EasyGui

You can download **easygui.py** or a zip file that contains **easygui.py**. To install it, you just have to put the file **easygui.py** in a place where Python can find it. Where is that?

The Python path

Python has a list of places on the hard drive where it looks for modules it can use. This can be a bit complicated, because it's different for Windows, Mac OS X, and Linux. But if you put **easygui.py** in the same place where Python itself is installed, Python will find it. So, on your hard drive, look for a folder called **Python25**, and put **easygui.py** in that folder.

Let's get GUI-ing

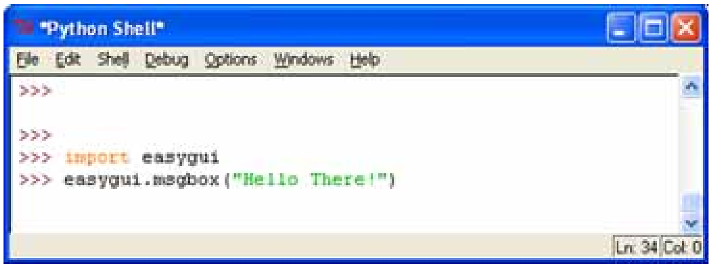
Start IDLE, and type the following in interactive mode:

```
>>> import easygui
```

This tells Python that you're going to use the EasyGui module. If you don't get an error message, then Python found the EasyGui module. If you do get an error message, or EasyGui doesn't seem to be working, go to the book's web site (www.helloworldbook.com) and you'll find some additional help.

Now, let's make a simple message box with an **OK** button:

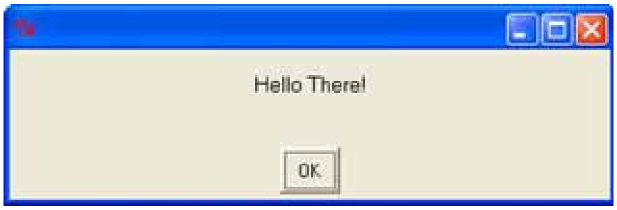
```
>>> easygui.msgbox("Hello There!")
```



```
>>>

>>>
>>> import easygui
>>> easygui.msgbox("Hello There!")
```

The EasyGui `msgbox()` function is used to create a message box. In most cases, the names of EasyGui functions are just shortened versions of the English words.

When you use `msgbox()`, you should see something that looks like this:



And if you click the **OK** button, the message box will close.

IDLE and EasyGui

Because of the way EasyGui and IDLE work, some people have had trouble using EasyGui from IDLE. If this doesn't work on your computer, you might have to run the EasyGui programs outside of IDLE. There are a number of ways to do this, but I'm going to tell you the easiest one.

If you installed Python using this book's installer, you also got a program called *SPE*, which stands for *Stani's Python Editor*. SPE is another way to edit and run your programs, just like IDLE. However, SPE doesn't have any problem working with EasyGui (as IDLE sometimes does).

You can start SPE and then open and edit Python files as you can with any other text editor. To run Python programs, use the **Tools > Run without arguments** command. You can use **CTRL-SHIFT-R** as a shortcut.

The only thing SPE doesn't have is a built-in shell that works. For interactive mode, or for text-based programs where the program asks the user for input and she has to type her response (like the number-guessing game from chapter 1), use **Tools > Run in Terminal without arguments**. The shortcut for this is **SHIFT-F9**. Or, stick with IDLE.

SPE is a good, easy-to-use editor for Python. It's free, open source software (just like Python). In fact, SPE is a Python program! If you prefer, you can use it for most of the examples in this book from now on. Give it a try and see if you like it.

GUI input

We just saw a kind of GUI output—a message box. But what about input? You can get input with EasyGui too.

When you ran the previous example in interactive mode, did you click on the **OK** button? If you did, you should have seen something like this in the shell or terminal or command window:

```
>>> import easygui
>>> easygui.msgbox("Hello there!")
'OK'
>>>
```

The `'OK'` part was Python and EasyGui telling you that the user clicked the **OK** button. EasyGui gives you back information to tell you what the user did in the GUI—what button she clicked, what she typed, etc. You can give this response a name (assign it to a variable). Try this:

```
>>> user_response = easygui.msgbox("Hello there!")
```

Click **OK** on the message box to close it. Then type this:

```
>>> print user_response
OK
>>>
```

Now the user's response, `OK,` has the variable name `user_response`. Let's look at a few other ways to get input with EasyGui.

The message box that we just saw is really just one example of something called a *dialog box*. Dialog boxes are GUI elements that are used to tell the user something or get some input from the user. The input might be a button click (like **OK**), or a filename, or some text (a string).

The EasyGui `msgbox` is just a dialog box with a message and a single button, **OK**. But we can have different kinds of dialog boxes with more buttons and other things.

Pick your flavor

We're going to use the example of choosing your favorite flavor of ice cream to look at some different ways to get input (the ice cream flavor) from the user with EasyGui.

Dialog box with multiple buttons

Let's make a dialog box (like a message box) with more than one button. The way to do this is with a *button box* (`buttonbox`). Let's make a program, rather than do it in interactive mode.

Start a new file in SPE (or another text editor if you're not using SPE). Type in the program in listing 6.1.

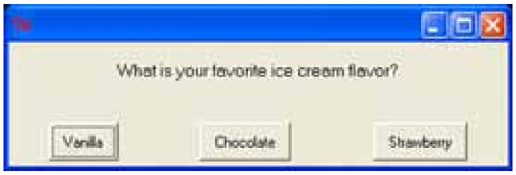
Listing 6.1 Getting input using buttons

```
import easygui
flavor = easygui.buttonbox("What is your favorite ice cream flavor?",
                  choices = ['Vanilla', 'Chocolate', 'Strawberry'] )

easygui.msgbox ("You picked " + flavor)
```

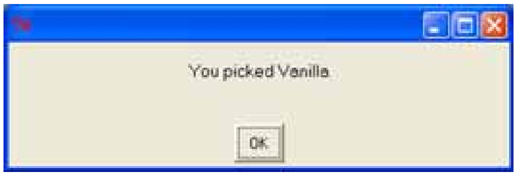
A list of choices

The part of the code in square brackets is called a *list*. We haven't talked about lists yet, but we'll learn all about them in chapter 12. For now, just type in the code so you can make the EasyGui program work. (Or, if you're really curious, you could skip ahead. . . .)

Save the file (I called mine **ice_cream1.py**) and run it. You should see this:


What is your favorite ice cream flavor?

And then, depending which flavor you click, you'll see something like this:


You picked Vanilla

How did this work? The label from whatever button the user clicked was the *input*. We assigned that input a variable name—in this case `flavor`. This is just like using `raw_input()`, except that the user doesn't type in the input, she just clicks a button. That's what GUIs are all about.

Choice box

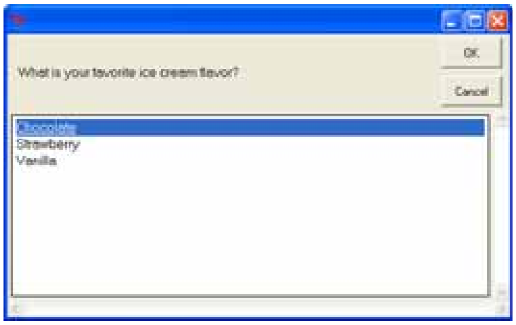
Let's try another way for the user to select a flavor. EasyGui has something called a *choice box* (`choicebox`), which presents a list of choices. The user picks one and then clicks the **OK** button.

To try this, we only need to make one small change to our program in listing 6.1: just change `buttonbox` to `choicebox`. The new version is shown in listing 6.2.

Listing 6.2 Getting input using a choice box

```
import easygui
flavor = easygui.choicebox("What is your favorite ice cream flavor?",
                  choices = ['Vanilla', 'Chocolate', 'Strawberry'] )
easygui.msgbox ("You picked " + flavor)
```

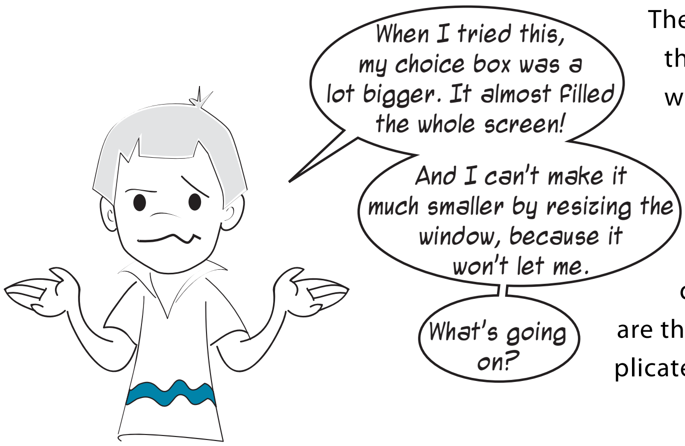
Save the program in listing 6.2 and run it. You should see something like this:


What is your favorite ice cream flavor?

Chocolate
Strawberry
Vanilla

After you click a flavor and then click **OK**, you'll see the same kind of message box as before. Notice that, in addition to clicking choices with the mouse, you can select a flavor with the up and down arrow keys on the keyboard.

If you click **Cancel**, the program will end, and you'll also see an error. That's because the last line of the program is expecting some text (like `Vanilla`), but if you click **Cancel**, it doesn't get any.


When I tried this, my choice box was a lot bigger. It almost filled the whole screen!
And I can't make it much smaller by resizing the window, because it won't let me.
What's going on?

The same thing happened to me. But that big choice box didn't fit very well in this book. So I cheated a bit! I modified **easygui.py** to let me make the choice box smaller so it would look nice in this book. It's not something you need to do, but if you really want to, here are the steps. I warn you, it's a bit complicated!

1 Find the section in the **easygui.py** file that starts with `def __choicebox` (around line 613 in my version of **easygui.py**). Remember that most editors, including SPE, show you the code line numbers somewhere near the bottom of the window.

2 About 30 lines down from that (around line 645), you'll see some lines that look like this:

```
root_width = int((screen_width * 0.8))
root_height = int((screen_height * 0.5))
```

3 Change the 0.8 to 0.4 and the 0.5 to 0.25. Save the changes to **easygui.py**. The next time you run the program, the choice box window will be smaller.

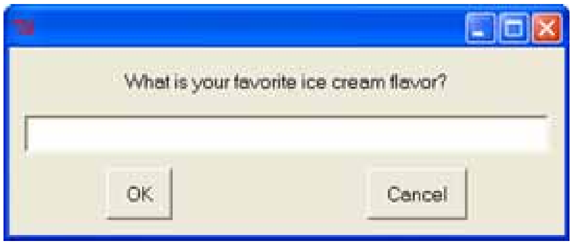
Text input

The examples in this chapter have let the user pick from a set of choices that you, as the programmer, provided. What if you want something more like `raw_input()`, where the user can type in text? That way, she can enter any flavor she wants. EasyGui has something called an enter box (`enterbox`) to do just that. Try the program in listing 6.3.

Listing 6.3 Getting input using an enter box

```
import easygui
flavor = easygui.enterbox("What is your favorite ice cream flavor?")
easygui.msgbox ("You entered " + flavor)
```

When you run it,
you should see something like this:


What is your favorite ice cream flavor?

And then, when you type in your favorite flavor and click **OK**, it'll be displayed in the message box, just like before.

This is just like `raw_input()`. It gets text (a string) from the user.

Default input

Sometimes when a user is entering information, there is a certain answer that is expected, common, or most likely to be entered. That is called a *default*. You might be able to save the user some typing by automatically entering the most common answer for her. Then, she'd only have to type an answer if she had a different input.

To put a default in an enter box, change your program to look like the one in listing 6.4.

Listing 6.4 How to make default arguments

```
import easygui
flavor = easygui.enterbox("What is your favorite ice cream flavor?",
                          default = 'Vanilla')
easygui.msgbox ("You entered " + flavor)
```

Here's the default

Now, when you run it, "Vanilla" is already entered in the enter box. You can delete it and enter anything you want, but if your favorite flavor is vanilla, you don't have to type anything, just click **OK**.

What about numbers?

If you want to enter a number in EasyGui, you can always use an enter box to get a string, and then create a number from it using `int()` or `float()` (as we did in chapter 4).

EasyGui also has something called an integer box (`integerbox`), which you can use for entering integers. You can set a lower and upper limit to the number that can be entered.

It doesn't let you enter floats (decimal numbers) though. To enter decimal numbers, you'd have to use an enter box, get the string, and then use `float()` to convert the string.

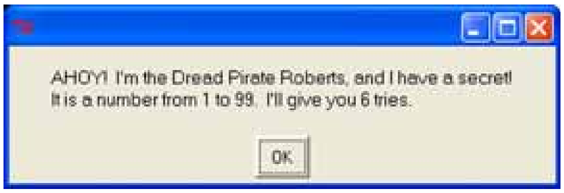
The number-guessing game . . . again

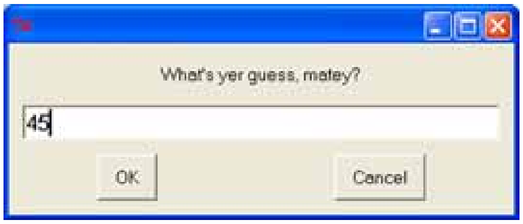
In chapter 1, we made a simple number-guessing program. Now let's try the same thing, but using EasyGui for the input and output. Listing 6.5 has the code.

Listing 6.5 Number-guessing game using EasyGui

```
import random, easygui

secret = random.randint(1, 100)     Picks a secret number
guess = 0
tries = 0

easygui.msgbox("""AHOY!  I'm the Dread Pirate Roberts, and I have a secret!
It is a number from 1 to 99.  I'll give you 6 tries.""")
                                                         Gets the player's guess
while guess != secret and tries < 6:
    guess = easygui.integerbox("What's yer guess, matey?")
    if not guess: break
    if guess < secret:                                              Allows up
        easygui.msgbox(str(guess) + " is too low, ye scurvy dog!")  to 6
    elif guess > secret:                                            guesses
        easygui.msgbox(str(guess) + " is too high, landlubber!")
    tries = tries + 1     Uses up one try

if guess == secret:                                                 Prints
    easygui.msgbox("Avast! Ye got it!  Found my secret, ye did!")   message
else:                                                               at end of
    easygui.msgbox("No more guesses!  Better luck next time, matey!") game
```

We still haven't learned how all the parts of this program work, but type it in and give it a try. You should see something like this when you run it:


AHOY! I'm the Dread Pirate Roberts, and I have a secret!
It is a number from 1 to 99. I'll give you 6 tries.


What's yer guess, matey?
45

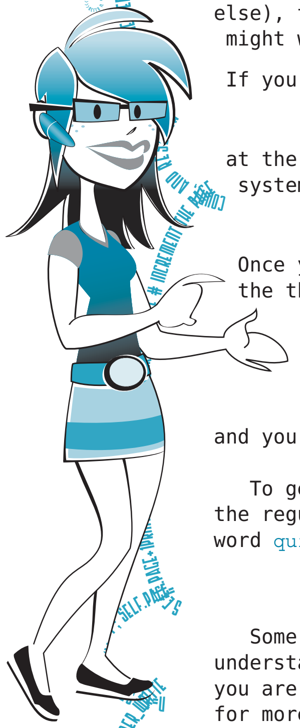
We'll be learning about `if`, `else`, and `elif` in chapter 7, and `while` in chapter 8. We'll learn about `random` in chapter 15, and we'll use it a lot more in chapter 23.

Other GUI pieces

EasyGui has a few other GUI pieces available, including a choice box that lets you pick multiple choices (instead of just one), and some special dialog boxes for getting filenames and so on. But the ones we have looked at are enough for now.

EasyGui makes generating some simple GUIs very easy, and it hides a lot of the complexity that is involved in GUIs so you don't have to worry about it. Later on, we'll look at another way to make GUIs that gives you a lot more flexibility and control.

If you want to find out more about EasyGui, you can go to the EasyGui home page at http://easygui.sourceforge.net.



Thinking like a (Python) programmer

If you want to find out more about something to do with Python, like EasyGui (or anything else), there is a built-in help system that you might want to try.

If you're in interactive mode, you can type

```
>>>help()
```

at the interactive prompt to get into the help system. The prompt will change to look like this:

```
help >
```

Once you're there, just type the name of the thing you want help with, like this:

```
help> time.sleep
```

or

```
help> easygui.msgbox
```

and you'll get some information.

To get out of the help system and back to the regular interactive prompt, just type the word `quit`:

```
help> quit
>>>
```

Some of the help is hard to read and understand, and you won't always find what you are looking for. But if you are looking for more information on something in Python, it's worth a try.

What did you learn?

In this chapter, you learned

- how to make simple GUIs with EasyGui.
- how to display messages using a message box: `msgbox`.
- how to get input using buttons, choice boxes, and text entry boxes: `buttonbox`, `choicebox`, `enterbox`, `integerbox`.
- how to set default input for a text box.
- how to use Python's built-in help system.

Test your knowledge

1. How do you bring up a message box with EasyGui?
2. How do you get a string (some text) as input using EasyGui?
3. How can you get an integer as input using EasyGui?
4. How can you get a float (decimal number) as input using EasyGui?
5. What's a default value? Give an example of something you might use it for.

Try it out

1. Try changing the temperature-conversion program from chapter 5 to use GUI input and output instead of `raw_input()` and `print`.
2. Write a program that asks for your name, then house number, then street, then city, then province/territory/state, then postal/zip code (all in EasyGui dialog boxes). The program should then display a mailing-style full address that looks something like this:

```
John Snead
28 Main Street
Akron, Ohio
12345
```

PROGRAMMING

Hello World!

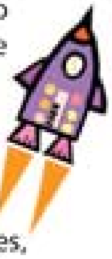
Computer Programming for Kids and Other Beginners

Warren Sande and Carter Sande

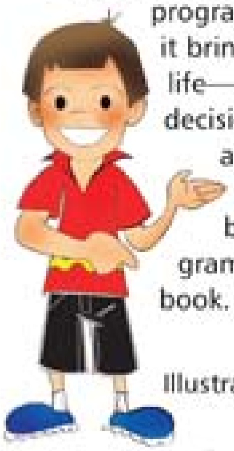
"Computer programming is a powerful tool for children to 'learn learning.' ... Children who engage in programming transfer that kind of learning to other things."

—Nicholas Negroponte
One Laptop Per Child Project, January 2008

Your computer won't respond when you yell at it, so why not talk to it in its own language? If you learn to program, you can do just that. You'll be able to do really cool things quickly, and even make your own games! Programming is fun!



Hello World! Computer Programming for Kids and Other Beginners is a wonderfully written introduction to programming. Using fun examples, it brings computing concepts to life—concepts like memory, loops, decisions, input and output, data, and graphics. It's written in a language a kid can follow, but anyone who wants to program a computer can use this book. Even adults!



Illustrated by Martin Murtonen

For online access to the authors, go to www.manning.com/HelloWorld

Hello World! uses Python, the programming language also chosen for the One Laptop Per Child Project. Readers with no previous knowledge of computing will be programming in no time at all.

WHAT'S INSIDE

- Explains everything in clear language—no "geek speak"
- Loaded with pictures, cartoons, and fun examples
- Complete set of practice questions and exercises
- Reviewed by professional educators, kid-tested, and parent-approved

Warren Sande is an Electronic Systems Engineer who uses Python and other computer languages in his daily work. His son, *Carter Sande*, is an elementary school student who loves computers, playing the piano, jumping on the trampoline, bike riding, and his little sister. His nickname at school is "Tech Support."

ISBN-13: 978-1933988498
ISBN-10: 1933988495
53499

MANNING US $34.99 / Can $34.99