PYTHON FOR KIDS

A PLAYFUL INTRODUCTION TO PROGRAMMING

JASON R. BRIGGS

no starch press

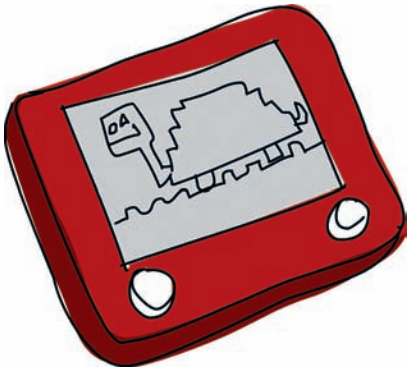
4
DRAWING WITH TURTLES

A *turtle* in Python is sort of like a turtle in the real world. We know a turtle as a reptile that moves around very slowly and carries its house on its back. In the world of Python, a turtle is a small, black arrow that moves slowly around the screen. Actually, considering that a Python turtle leaves a trail as it moves around the screen, it's actually less like a turtle and more like a snail or a slug.

The turtle is a nice way to learn some of the basics of computer graphics, so in this chapter, we'll use a Python turtle to draw some simple shapes and lines.

USING PYTHON'S TURTLE MODULE

A *module* in Python is a way of providing useful code to be used by another program (among other things, the module can contain functions we can use). We'll learn more about modules in Chapter 7. Python has a special module called `turtle` that we can use to learn how computers draw pictures on a screen. The `turtle` module is a way of programming vector graphics, which is basically just drawing with simple lines, dots, and curves.



Let's see how the turtle works. First, start the Python shell by clicking the desktop icon (or if you're using Ubuntu, select **Applications ▸ Programming ▸ IDLE**). Next, tell Python to use the turtle by importing the `turtle` module, as follows:

```
>>> import turtle
```

Importing a module tells Python that you want to use it.

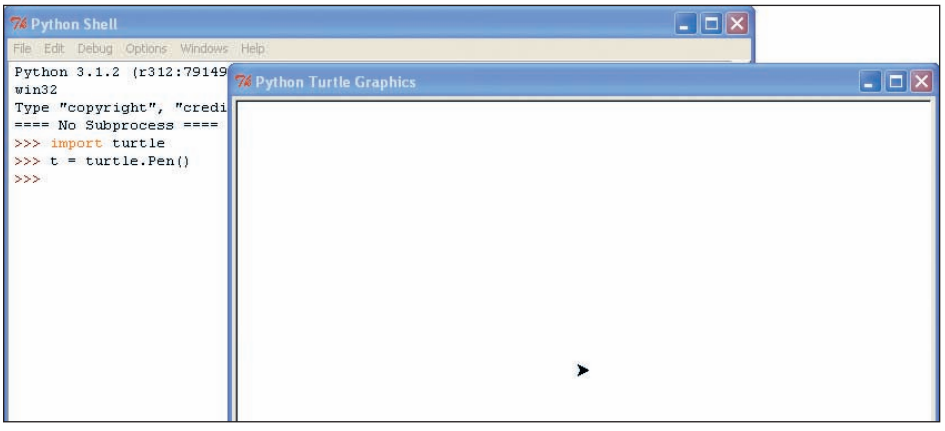
NOTE *If you're using Ubuntu and you get an error at this point, you might need to install* `tkinter`*. To do so, open the Ubuntu Software Center and enter* python-tk *in the search box. "Tkinter – Writing Tk Applications with Python" should appear in the window. Click* ***Install*** *to install this package.*

CREATING A CANVAS

Now that we have imported the `turtle` module, we need to create a canvas—a blank space to draw on, like an artist's canvas. To do so, we call the function `Pen` from the `turtle` module, which automatically creates a canvas. Enter this into the Python shell:

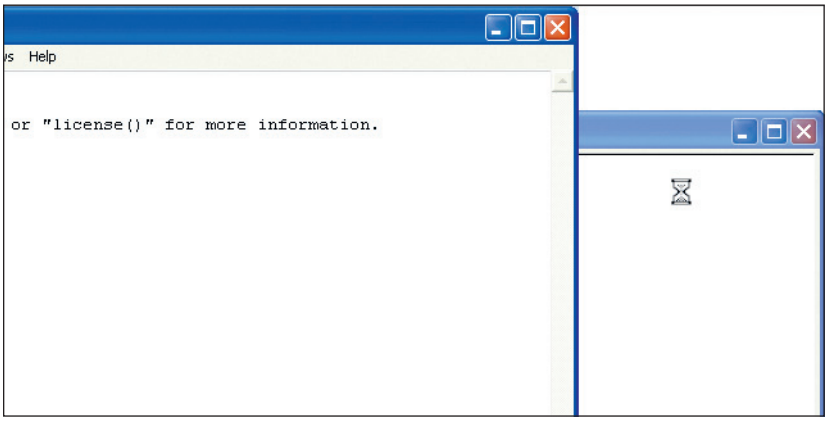
```
>>> t = turtle.Pen()
```

You should see a blank box (the canvas), with an arrow in the center, something like this:



The arrow in the middle of the screen is the turtle, and you're right—it isn't very turtle-like.

If the Turtle window appears behind the Python Shell window, you may find that it doesn't seem to be working properly. When you move your mouse over the Turtle window, the cursor turns into an hourglass, like this:

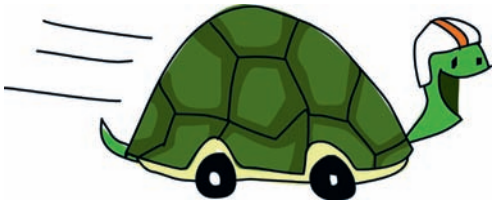


This could happen for several reasons: you haven't started the shell from the icon on your desktop (if you're using Windows or a Mac), you clicked IDLE (Python GUI) in the Windows Start menu,

or IDLE isn't installed correctly. Try exiting and restarting the shell from the desktop icon. If that fails, try using the Python console instead of the shell, as follows:

- In Windows, select **Start ▸All Programs**, and then in the **Python 3.2** group, click **Python (command line)**.
- In Mac OS X, click the Spotlight icon at the top-right corner of the screen and enter *Terminal* in the input box. Then enter *python* when the terminal opens.
- In Ubuntu, open the terminal from your **Applications** menu and enter *python*.

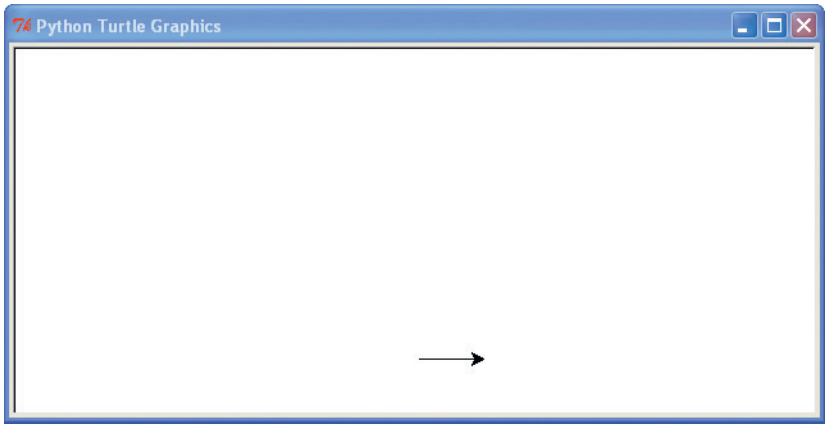
MOVING THE TURTLE



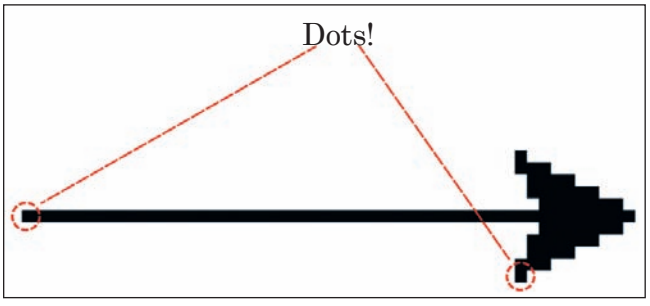
You send instructions to the turtle by using functions available on the variable `t` we just created, similar to using the `Pen` function in the `turtle` module. For example, the `forward` instruction tells the turtle to move forward. To tell the turtle to advance 50 pixels, enter the following command:

```
>>> t.forward(50)
```

You should see something like this:



The turtle has moved forward 50 pixels. A *pixel* is a single point on the screen—the smallest element that can be represented. Everything you see on your computer monitor is made up of pixels, which are tiny, square dots. If you could zoom in on the canvas and the line drawn by the turtle, you would be able to see that the arrow representing the turtle's path is just a bunch of pixels. That's simple computer graphics.



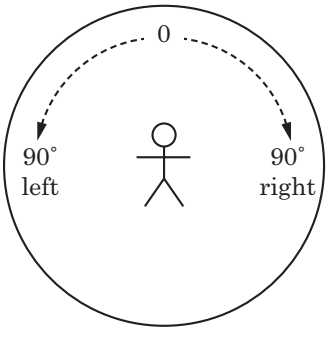
Now we'll tell the turtle to turn left 90 degrees with the following command:

```
>>> t.left(90)
```

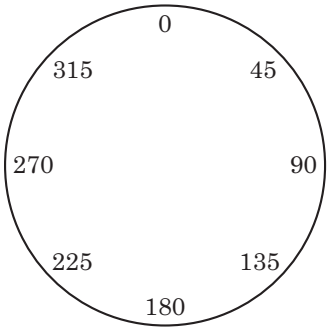
If you haven't learned about degrees yet, here's how to think about them. Imagine that you're standing in the center of a circle.

- The direction you're facing is 0 degrees.
- If you hold out your left arm, that's 90 degrees left.
- If you hold out your right arm, that's 90 degrees right.

You can see this 90-degree turn to the left or right here:

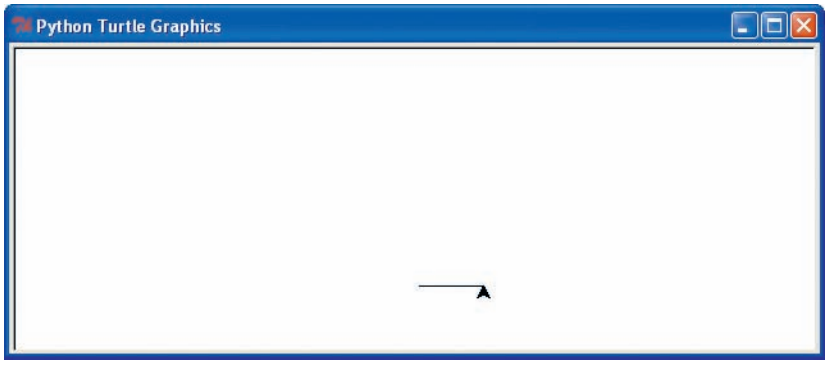


If you continue around the circle to the right from where your right arm is pointing, 180 degrees is directly behind you, 270 degrees is the direction your left arm is pointing, and 360 degrees is back where you started; degrees go from 0 to 360. The degrees in a full circle, when turning to the right, can be seen here in 45-degree increments:



When Python's turtle turns left, it swivels around to face the new direction (just as if you turned your body to face where your arm is pointing 90 degrees left).

The `t.left(90)` command points the arrow up (since it started by pointing to the right):



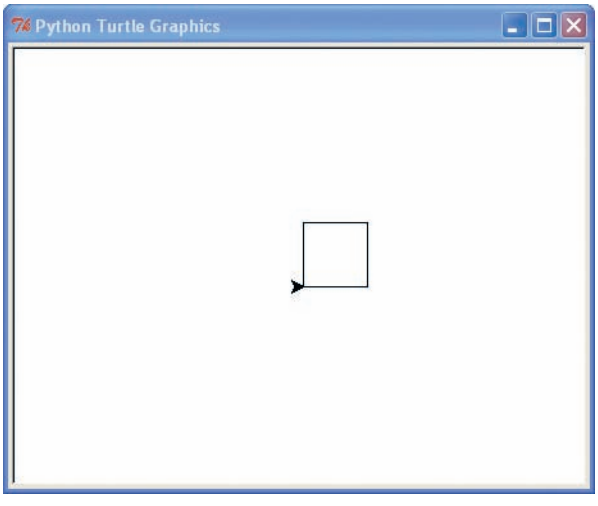
NOTE *When you call `t.left(90)`, it's the same as calling `t.right(270)`. This is also true of calling `t.right(90)`, which is the same as `t.left(270)`. Just imagine that circle and follow along with the degrees.*

Now we'll draw a square. Add the following code to the lines you've already entered:

```
>>> t.forward(50)
>>> t.left(90)
```

```
>>> t.forward(50)
>>> t.left(90)
>>> t.forward(50)
>>> t.left(90)
```

Your turtle should have drawn a square and should now be facing in the same direction it started:



To erase the canvas, enter `reset`. This clears the canvas and puts the turtle back at its starting position.

```
>>> t.reset()
```

You can also use `clear`, which just clears the screen and leaves the turtle where it is.

```
>>> t.clear()
```

We can also turn our turtle `right` or move it `backward`. We can use `up` to lift the pen off the page (in other words, tell the turtle to stop drawing), and `down` to start drawing. These functions are written in the same way as the others we've used.

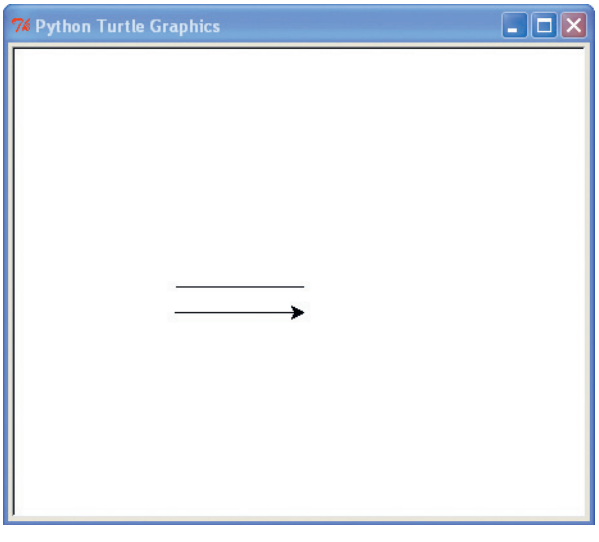
Let's try another drawing using some of these commands. This time, we'll have the turtle draw two lines. Enter the following code:

```
>>> t.reset()
>>> t.backward(100)
>>> t.up()
>>> t.right(90)
```

```
>>> t.forward(20)
>>> t.left(90)
>>> t.down()
>>> t.forward(100)
```

First, we reset the canvas and move the turtle back to its starting position with `t.reset()`. Next, we move the turtle backward 100 pixels with `t.backward(100)`, and then use `t.up()` to pick up the pen and stop drawing.

Then, with the command `t.right(90)`, we turn the turtle right 90 degrees to point down, toward the bottom of the screen, and with `t.forward(20)`, we move forward 20 pixels. Nothing is drawn because of the use of `up` command on the third line. We turn the turtle left 90 degrees to face right with `t.left(90)`, and then with the `down` command, we tell the turtle to put the pen back down and start drawing again. Finally, we draw a line forward, parallel to the first line we drew, with `t.forward(100)`. The two parallel lines we've drawn end up looking like this:



WHAT YOU LEARNED

In this chapter, you learned how to use Python's `turtle` module. We drew some simple lines, using `left` and `right` turns and `forward` and `backward` commands. You found out how to stop the turtle from drawing using `up`, and start drawing again with `down`. You also discovered that the turtle turns by degrees.

PROGRAMMING PUZZLES

Try drawing some of the following shapes with the turtle. The answers can be found at *http://python-for-kids.com/*.

#1: A RECTANGLE

Create a new canvas using the `turtle` module's `Pen` function and then draw a rectangle.

#2: A TRIANGLE

Create another canvas, and this time, draw a triangle. Look back at the diagram of the circle with the degrees ("Moving the Turtle" on page 46) to remind yourself which direction to turn the turtle using degrees.

#3: A BOX WITHOUT CORNERS

Write a program to draw the four lines shown here (the size isn't important, just the shape):