**Name: Session:**

**Programming I**

**Lab Exercise 10.2.2019**

**Graphics: Simple Drawings**

There are times when you may want to include simple drawings such as figures, graphs, or charts in your programs. Although the Python library provides a module for creating full graphical applications, we are going to use a simple graphics library found in the file ***graphics.py***. It is critical that this file either be stored in your development environment or in the same folder as your graphics application. For simplicity we will do the later.

To help you create simple drawings, this graphics module is a simplified version of Python’s more complex library module. The module code and usage instructions are available within the source code. In this exercise, you will learn all about this module, and how to use it for creating simple drawings that consist of basic geometric shapes and text.

**Creating a Window**

A graphical application shows information inside a **window** on the desktop with a

rectangular area and a title bar. In the graphics module, this window is called a *graphics window*.

To create a graphical application using the graphics module, carry out the following:

1. Import the GraphicsWindow class:

from graphics import GraphicsWindow

We will create a single object of the GraphicsWindow class and call methods on it.

1. Create a graphics window:

win = GraphicsWindow()

The new window will automatically be shown on the desktop and contain a

canvas that is 400 pixels wide by 400 pixels tall. To create a graphics window

with a canvas that is of a specific size, you can specify the width and height of

the canvas as arguments:

win = GraphicsWindow(500, 500)

When a graphics window is created, the object representing the window is

returned and must be stored in a variable, as it will be needed in the following

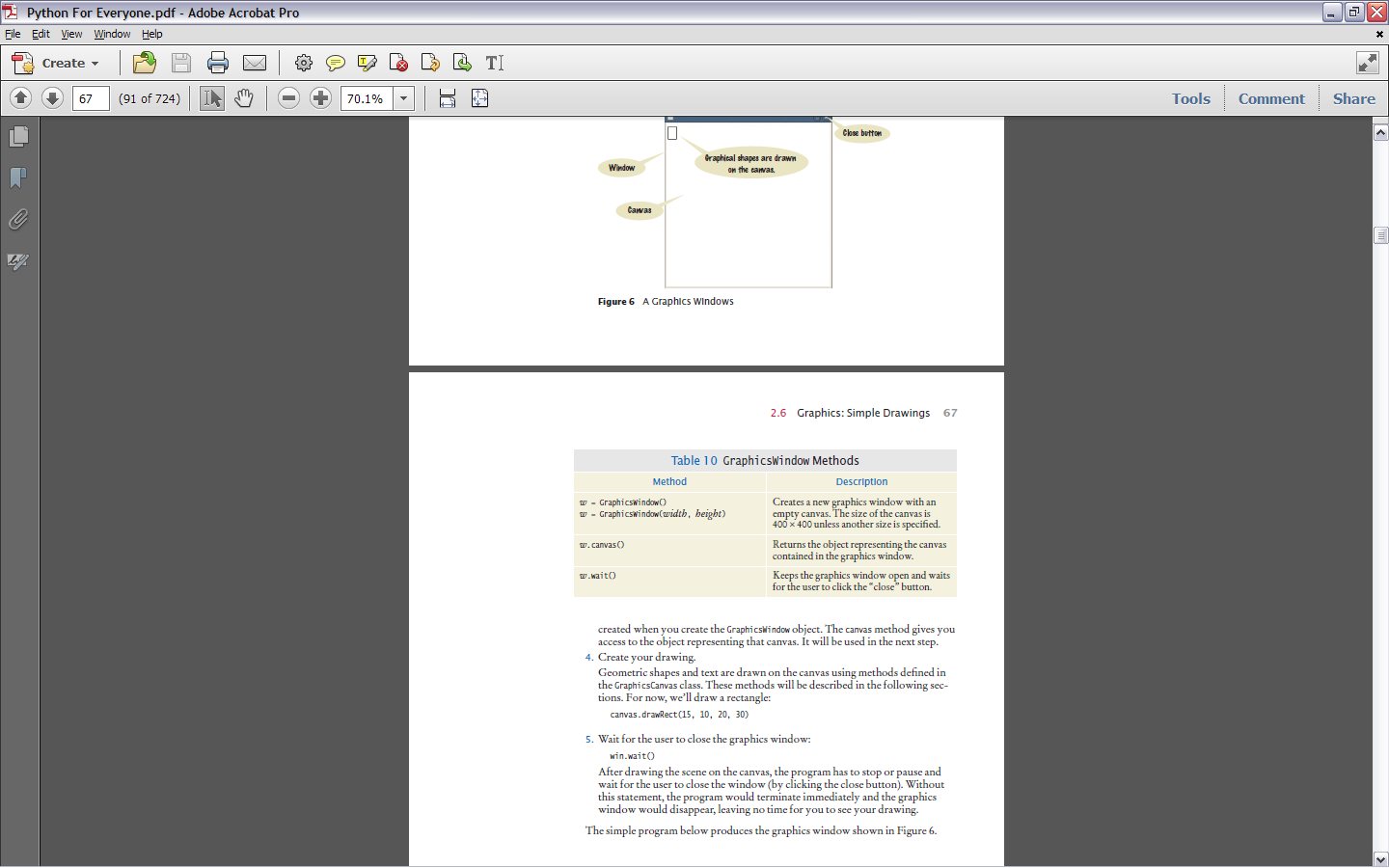
steps. Several methods that can be used with a GraphicsWindow object are shown

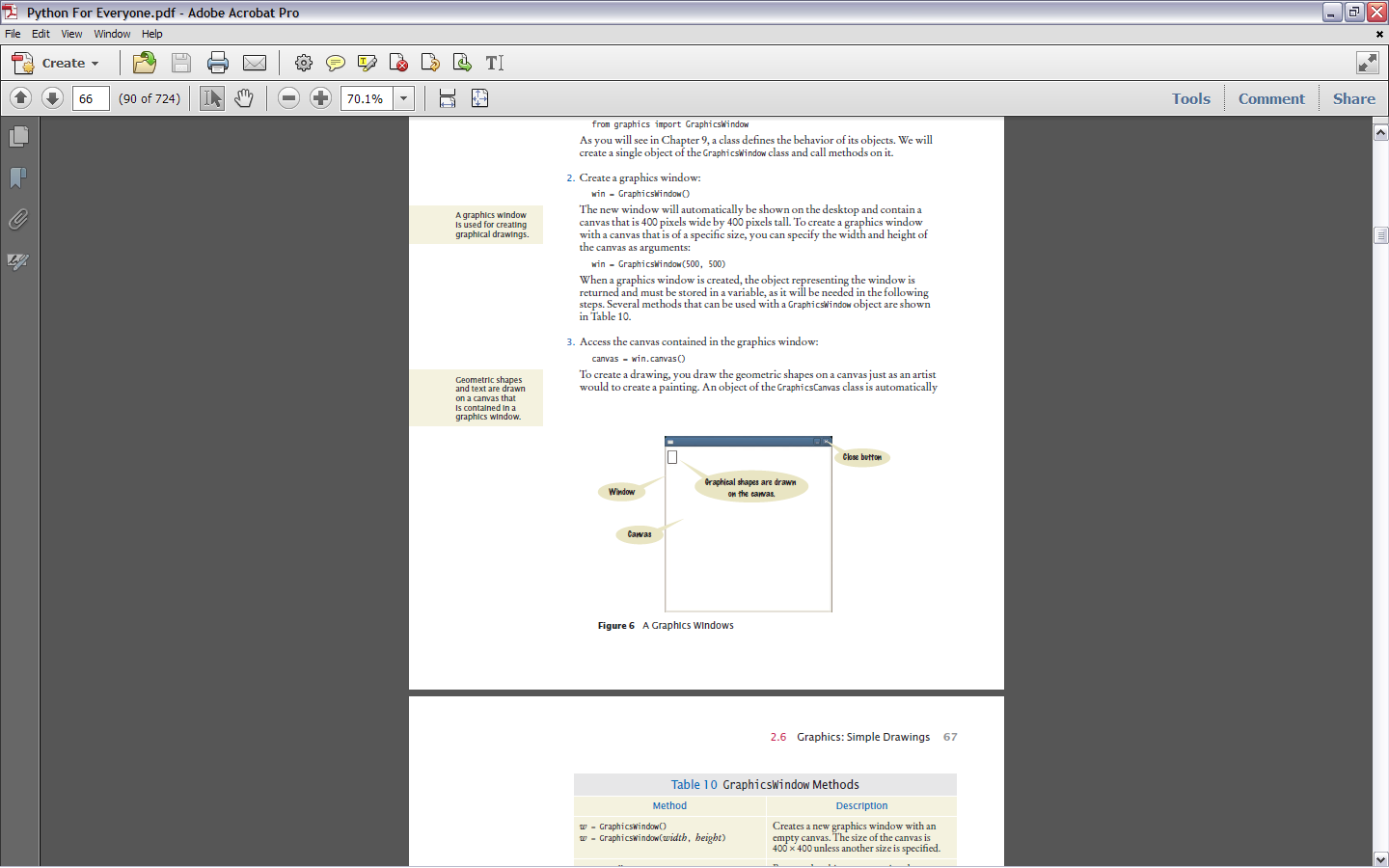
in Table 10.

1. Access the canvas contained in the graphics window:

canvas = win.canvas()

To create a drawing, you draw the geometric shapes on a canvas just as an artist would to create a painting. An object of the GraphicsCanvas class is automatically created when you create the GraphicsWindow object. The canvas method gives you access to the object representing that canvas. It will be used in the next step.





A Graphics Window

4. Create your drawing.

Geometric shapes and text are drawn on the canvas using methods defined in the GraphicsCanvas class. These methods will be described in the following sections. For now, we’ll draw a rectangle:

canvas.drawRect(15, 10, 20, 30)

5. Wait for the user to close the graphics window:

win.wait()

After drawing the scene on the canvas, the program has to stop or pause and wait for the user to close the window (by clicking the close button). Without this statement, the program would terminate immediately and the graphics window would disappear, leaving no time for you to see your drawing.

**window.py**

##

# This program creates a graphics window with a rectangle. It provides the

# template used with all of the graphical programs used in the book.

#

from graphics import GraphicsWindow

# Create the window and access the canvas.

win = GraphicsWindow()

canvas = win.canvas()

# Draw on the canvas.

canvas.drawRect(5, 10, 20, 30)

# Wait for the user to close the window.

win.wait()

**Lines and Polygons**

To draw a shape on the canvas, you call one of the “draw” methods defined for a

canvas. The call

canvas.drawLine(x1, y1, x2, y2)

draws a line on the canvas between the points (*x*1, *y*1) and (*x*2, *y*2). The call

canvas.drawRect(x, y, width, height)

draws a rectangle that has its upper-left corner positioned at (*x*, *y*) and the given width

and height.

Geometric shapes and text are drawn on a canvas by specifying points in the two-dimensional

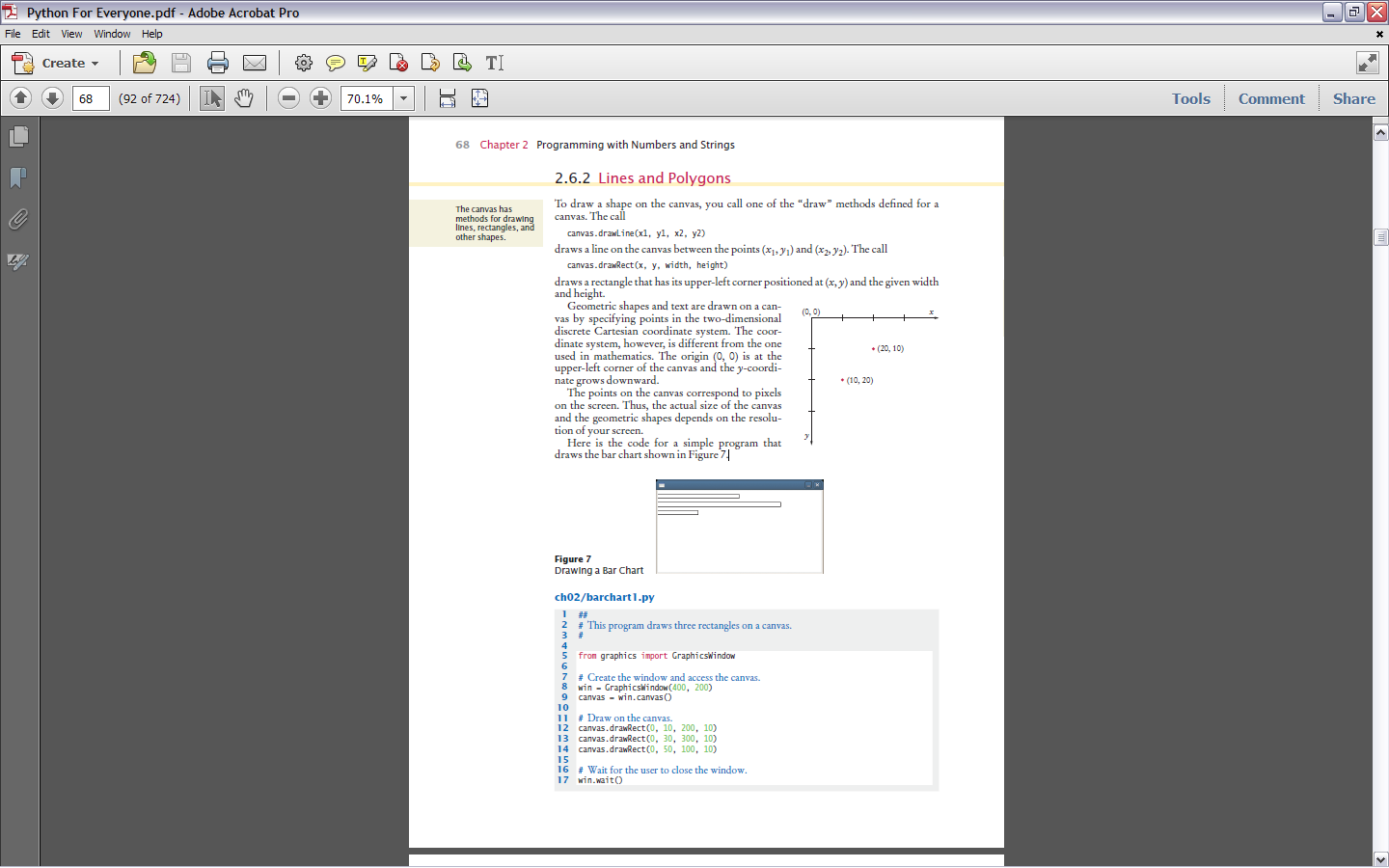
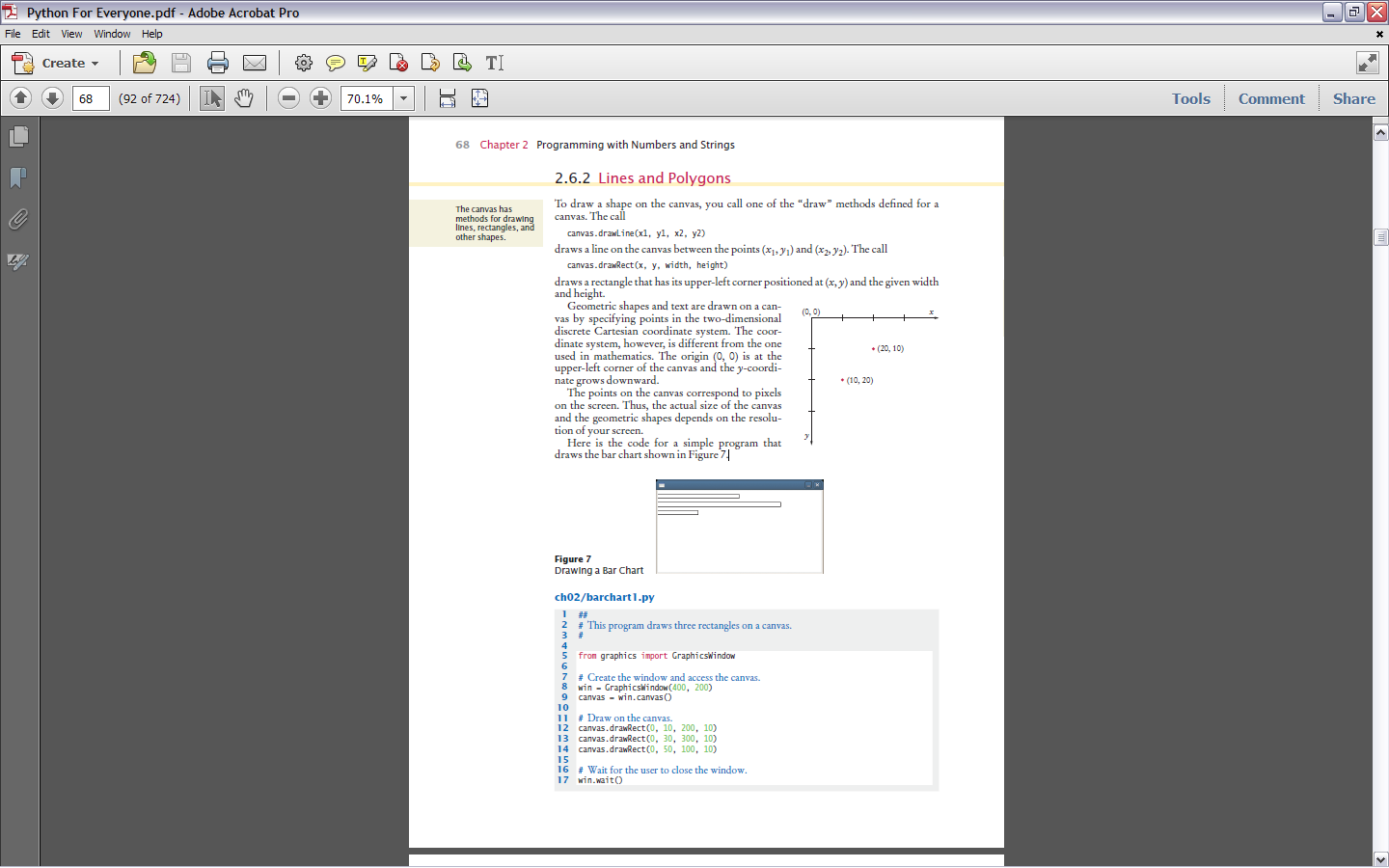
discrete Cartesian coordinate system. The coordinate system, however, is different from the one

used in mathematics. The origin (0, 0) is at the upper-left corner of the canvas and the *y*-coordinate

grows downward (see below).

The points on the canvas correspond to pixels on the screen. Thus, the actual size of the canvas

and the geometric shapes depends on the resolution of your screen. Here is the code for a simple program that draws the bar chart shown below.

**barchart1.py**

# This program draws three rectangles on a canvas.

from graphics import GraphicsWindow

# Create the window and access the canvas.

win = GraphicsWindow(400, 200)

canvas = win.canvas()

# Draw on the canvas.

canvas.drawRect(0, 10, 200, 10)

canvas.drawRect(0, 30, 300, 10)

canvas.drawRect(0, 50, 100, 10)

# Wait for the user to close the window.

win.wait()

**Filled Shapes and Color**

The canvas stores the drawing parameters (the current color, font, line width, and soon) that are used for d rawing shapes and text. When you first start drawing on a canvas, all shapes are drawn using a black pen.

To change the pen color, use one of the method calls,

canvas.setOutline(red, green, blue)

canvas.setOutline(colorName)

The method arguments can be integer values between 0 and 255 that specify a color value, or one of the strings describing a color in Table 11.

For example, to draw a red rectangle, call

canvas.setOutline(255, 0, 0)

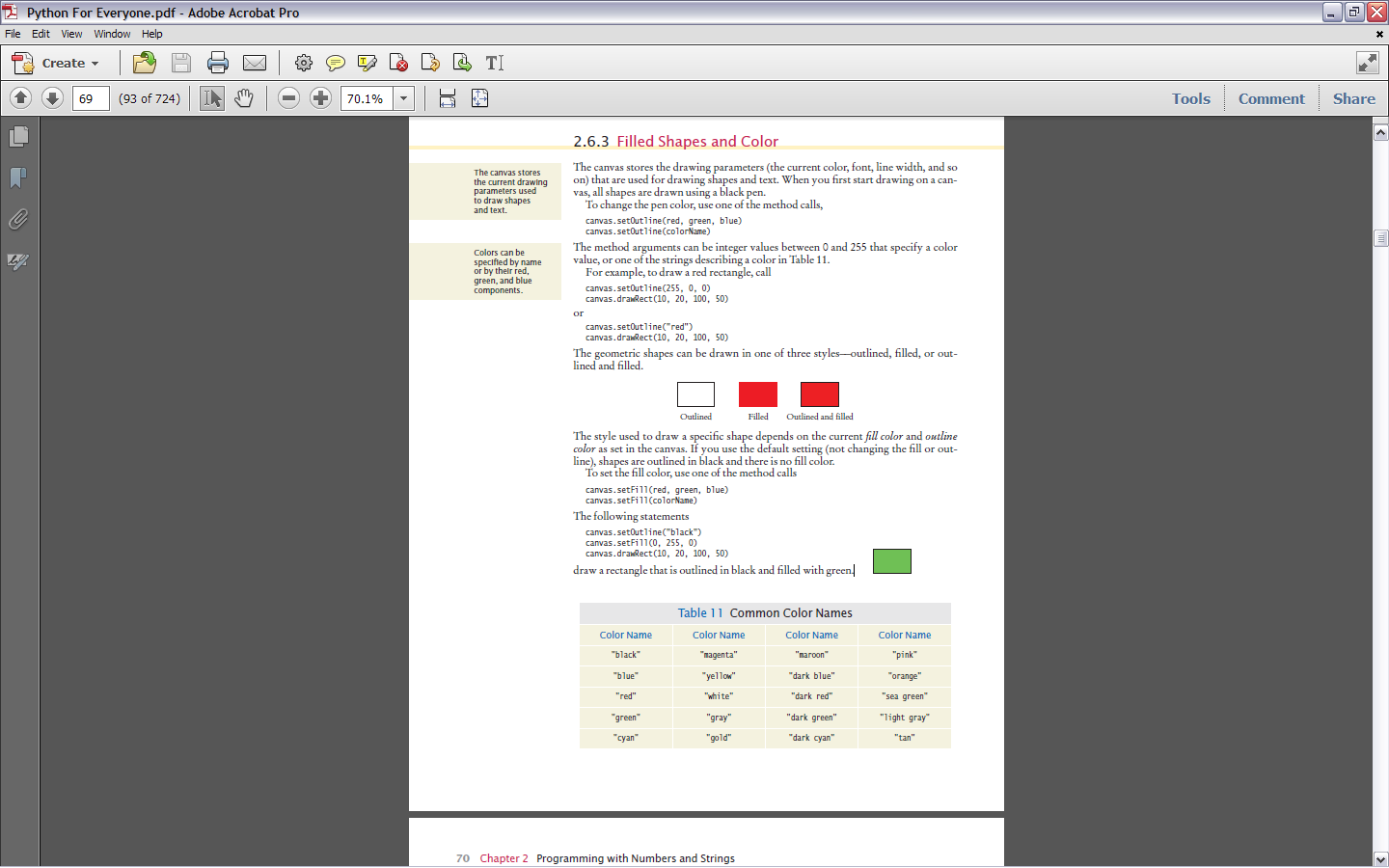
canvas.drawRect(10, 20, 100, 50)

or

canvas.setOutline("red")

canvas.drawRect(10, 20, 100, 50)

The geometric shapes can be drawn in one of three styles––outlined, filled, or outlined and filled.



The style used to draw a specific shape depends on the current *fill color* and *outline color* as set in the canvas. If you use the default setting (not changing the fill or outline),shapes are outlined in black and there is no fill color.

To set the fill color, use one of the method calls

canvas.setFill(red, green, blue)

canvas.setFill(colorName)

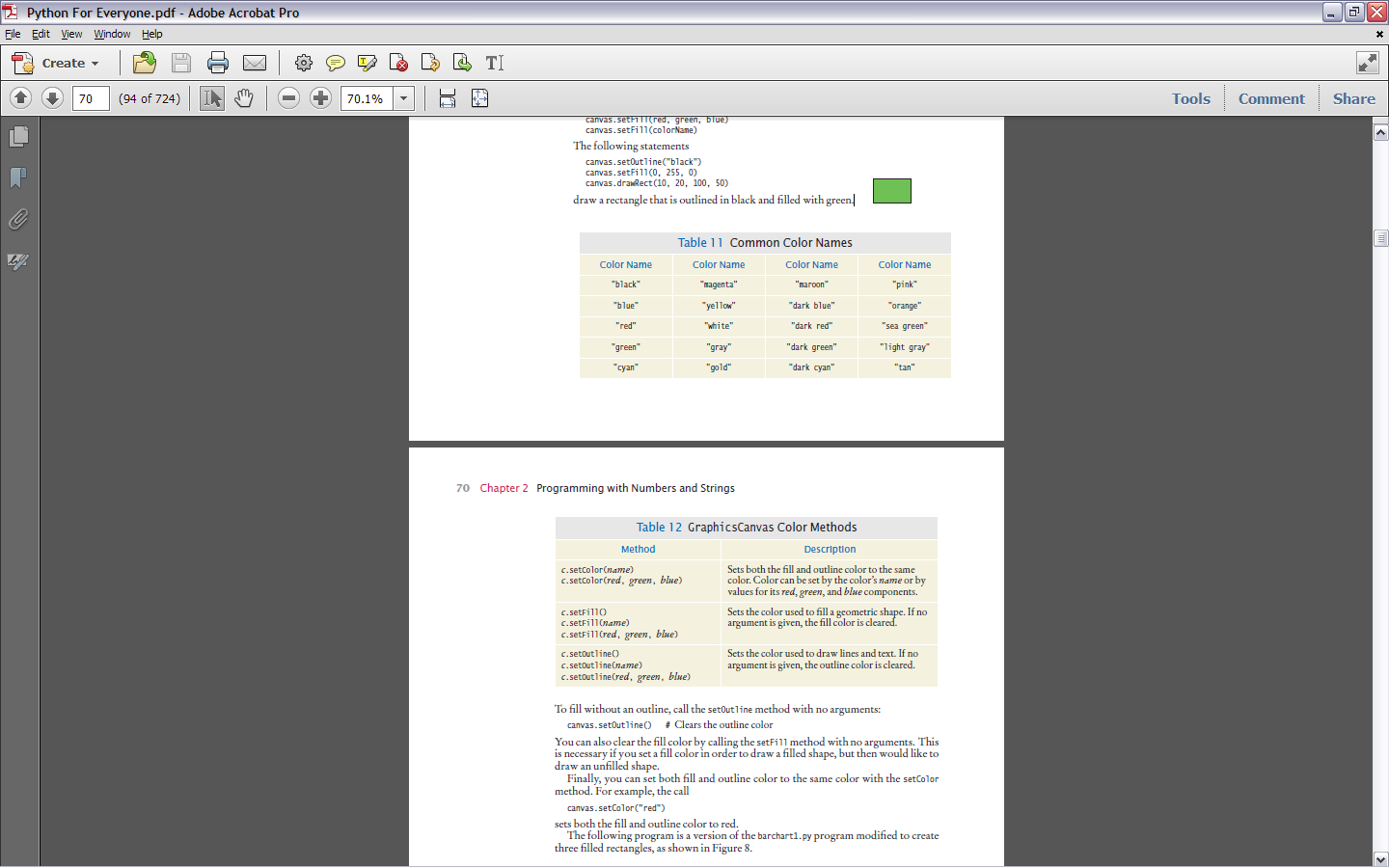
The following statements

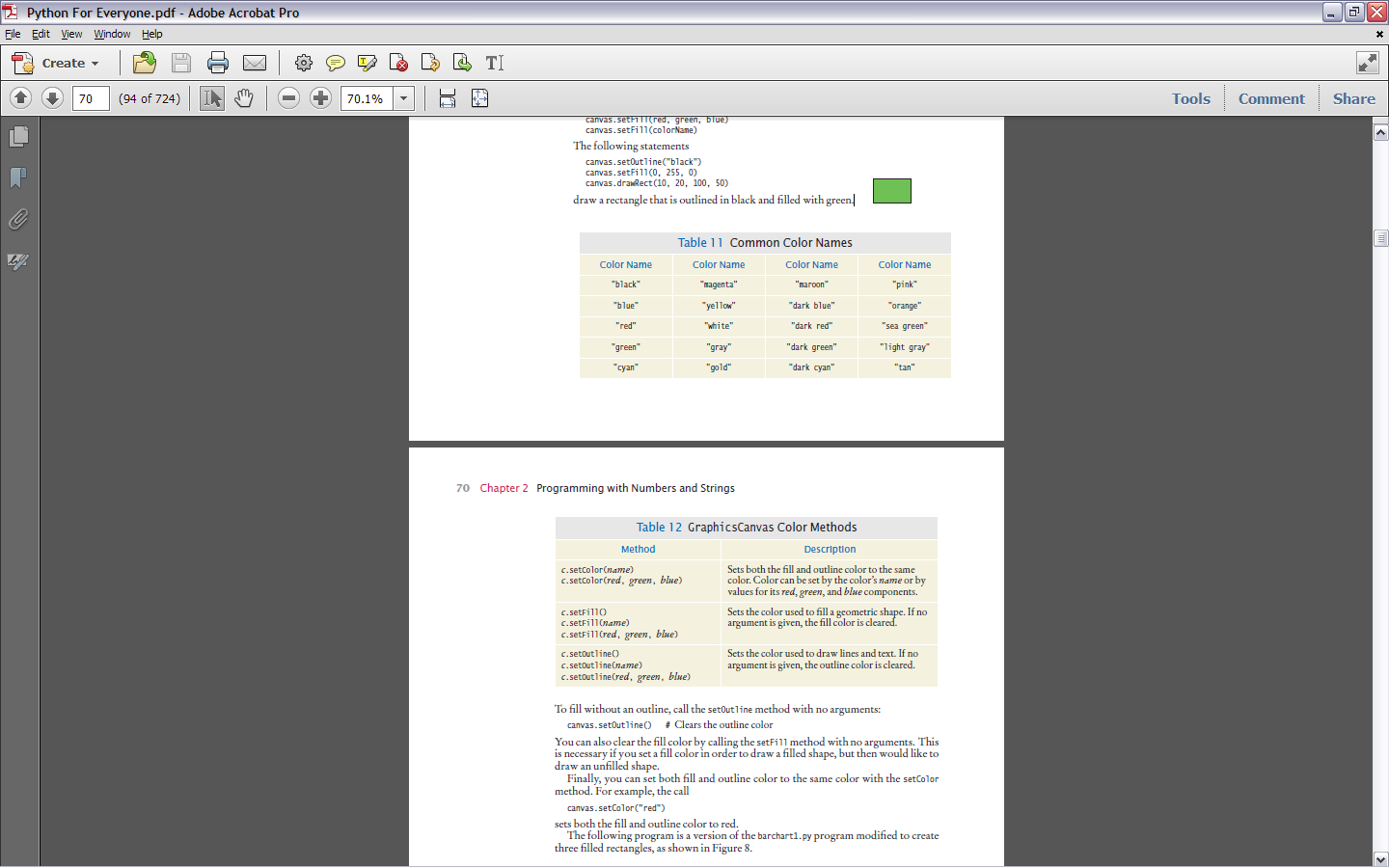
canvas.setOutline("black")

canvas.setFill(0, 255, 0)

canvas.drawRect(10, 20, 100, 50)

draw a rectangle that is outlined in black and filled with green.





To fill without an outline, call the setOutline method with no arguments:

canvas.setOutline() # Clears the outline color

You can also clear the fill color by calling the setFill method with no arguments. This is necessary if you set a fill color in order to draw a filled shape, but then would like to draw an unfilled shape. Finally, you can set both fill and outline color to the same color with the setColor method. For example, the call

canvas.setColor("red")

sets both the fill and outline color to red. The following program is a version of the barchart1.py program modified to create three filled rectangles, as shown below.

**barchart2.py**

##

# This programs draws three colored rectangles on a canvas.

from graphics import GraphicsWindow

# Create the window and access the canvas.

win = GraphicsWindow(400, 200)

canvas = win.canvas()

# Draw on the canvas.

canvas.setColor("red")

canvas.drawRect(0, 10, 200, 10)

canvas.setColor("green")

canvas.drawRect(0, 30, 300, 10)

canvas.setColor("blue")

canvas.drawRect(0, 50, 100, 10)

# Wait for the user to close the

# window.

win.wait()

**Ovals, Circles, and Text**

Now that you’ve learned how to draw lines and rectangles, let’s turn to additional

graphical elements. To draw an oval, you specify its *bounding box* (see Figure 9) in the same way that

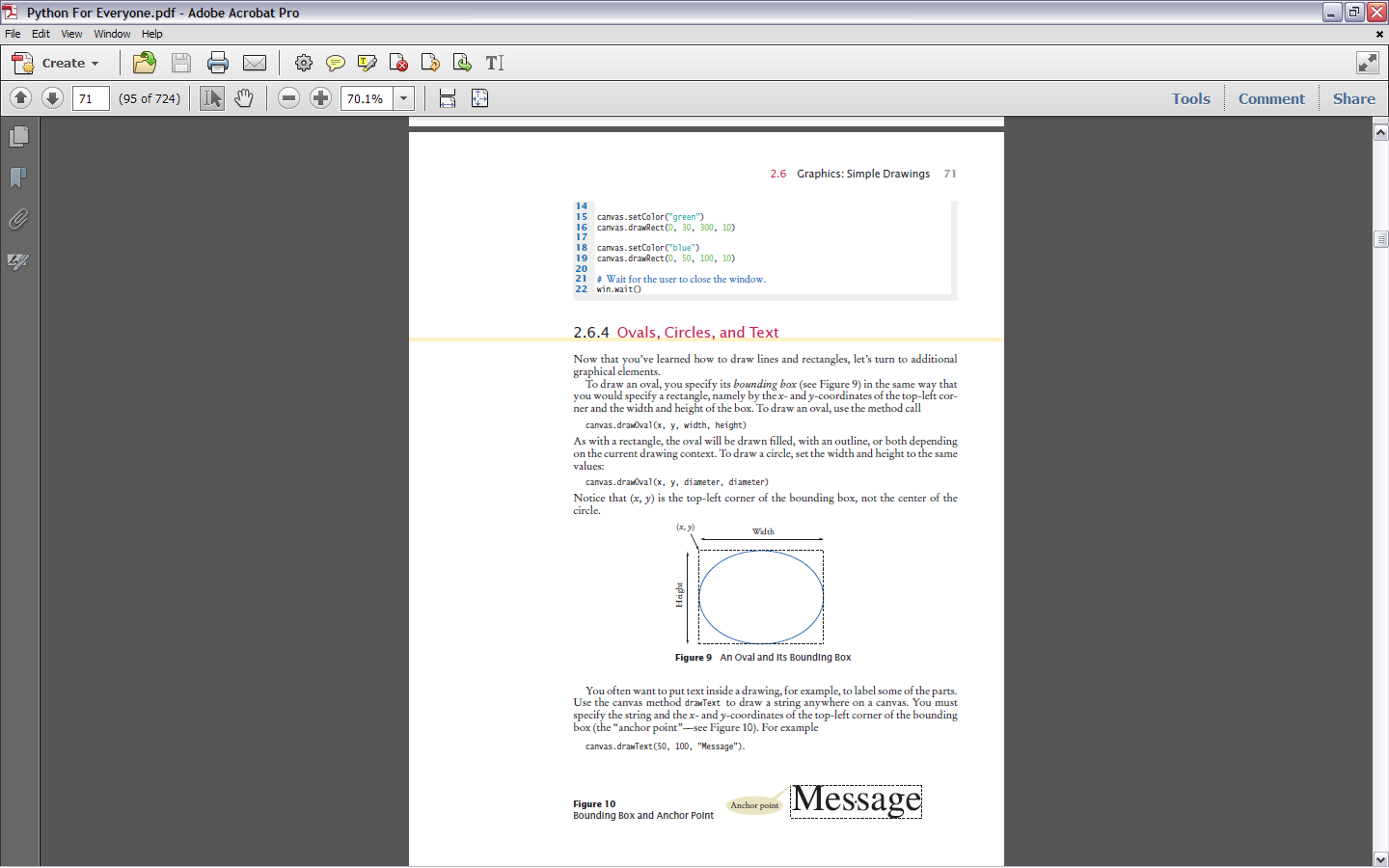
you would specify a rectangle, namely by the *x*- and *y*-coordinates of the top-left corner and the width and height of the box. To draw an oval, use the method call

canvas.drawOval(x, y, width, height)

As with a rectangle, the oval will be drawn filled, with an outline, or both depending on the current drawing context. To draw a circle, set the width and height to the same values:

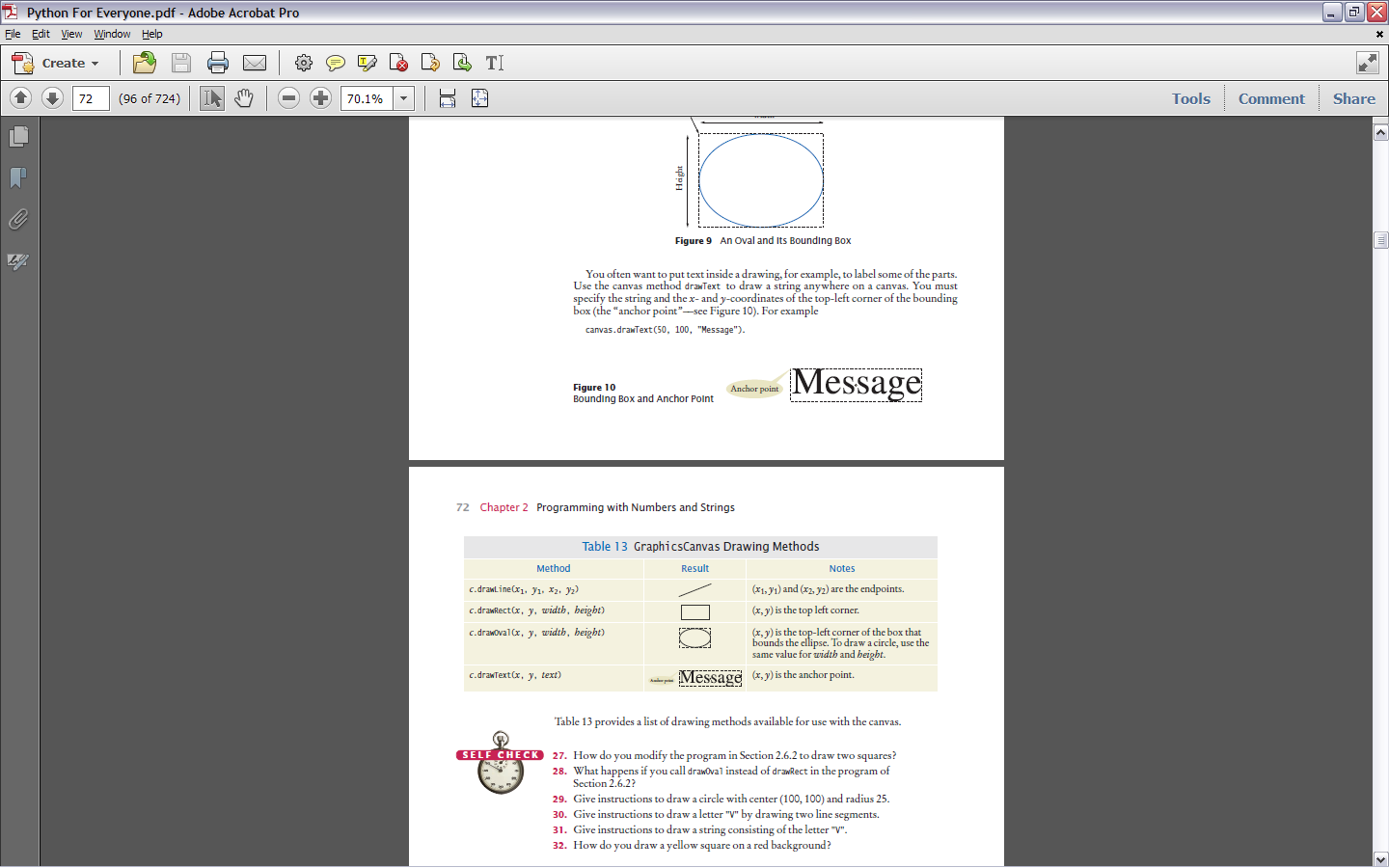
canvas.drawOval(x, y, diameter, diameter)

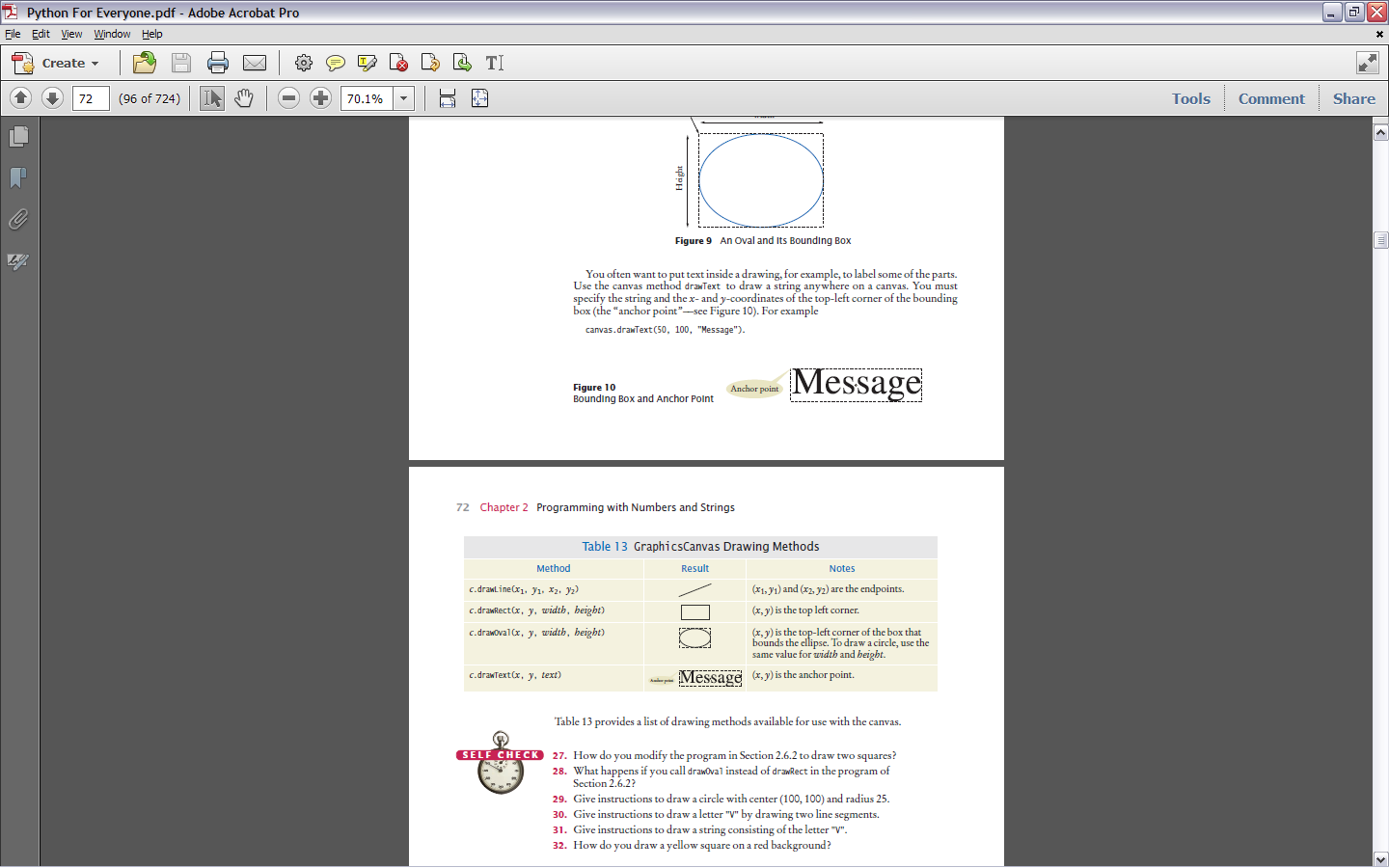
Notice that (*x*, *y*) is the top-left corner of the bounding box, not the center of the circle.



You often want to put text inside a drawing, for example, to label some of the parts. Use the canvas method drawText to draw a string anywhere on a canvas. You must specify the string and the *x*- and *y*-coordinates of the top-left corner of the bounding box (the “anchor point”––see below). For example

canvas.drawText(50, 100, "Message").





**Exercises (make sure you have the graphics.py module in your program folder)**

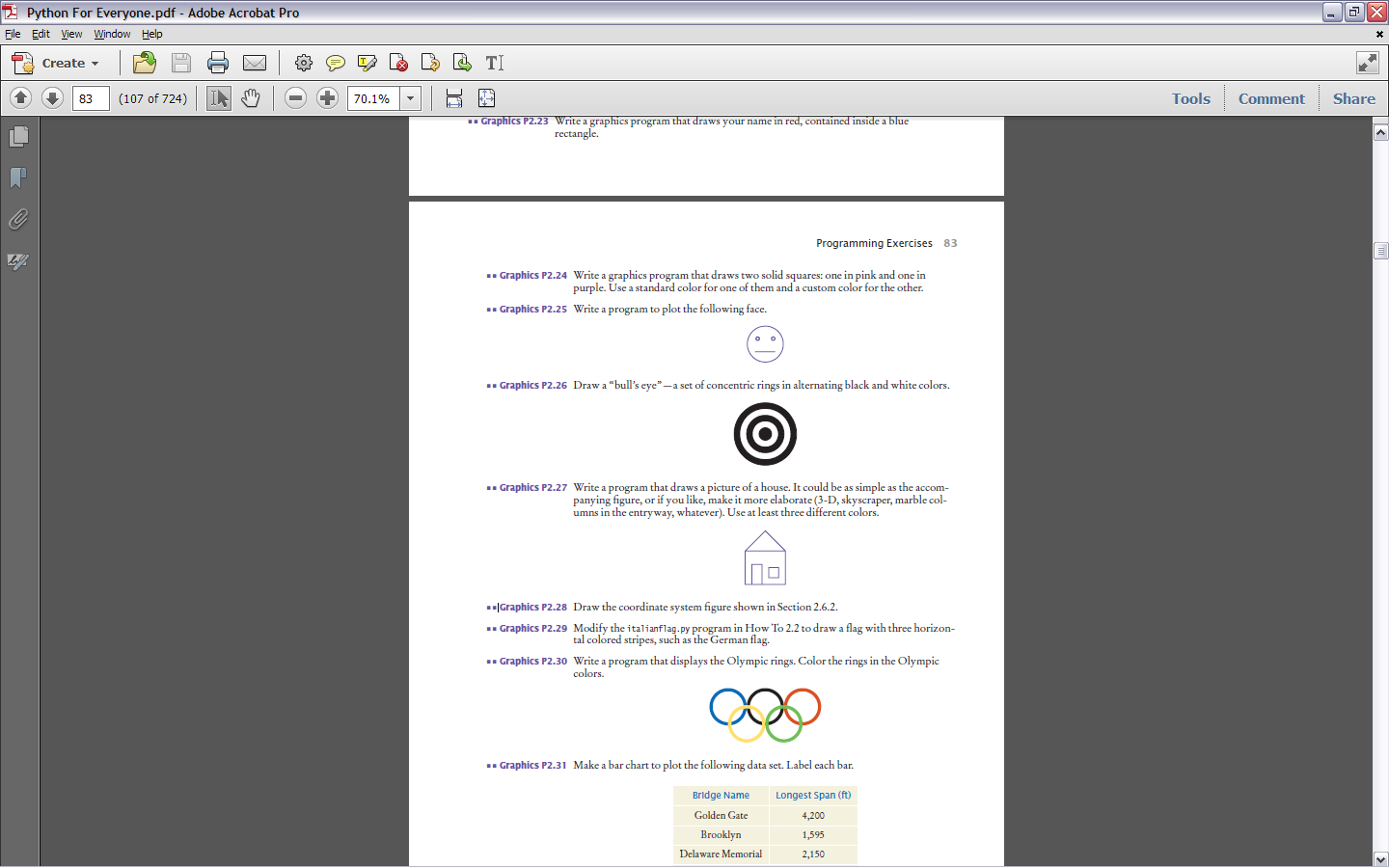
1. Write a graphics program that draws your name in Red, contained inside a Blue

rectangle.

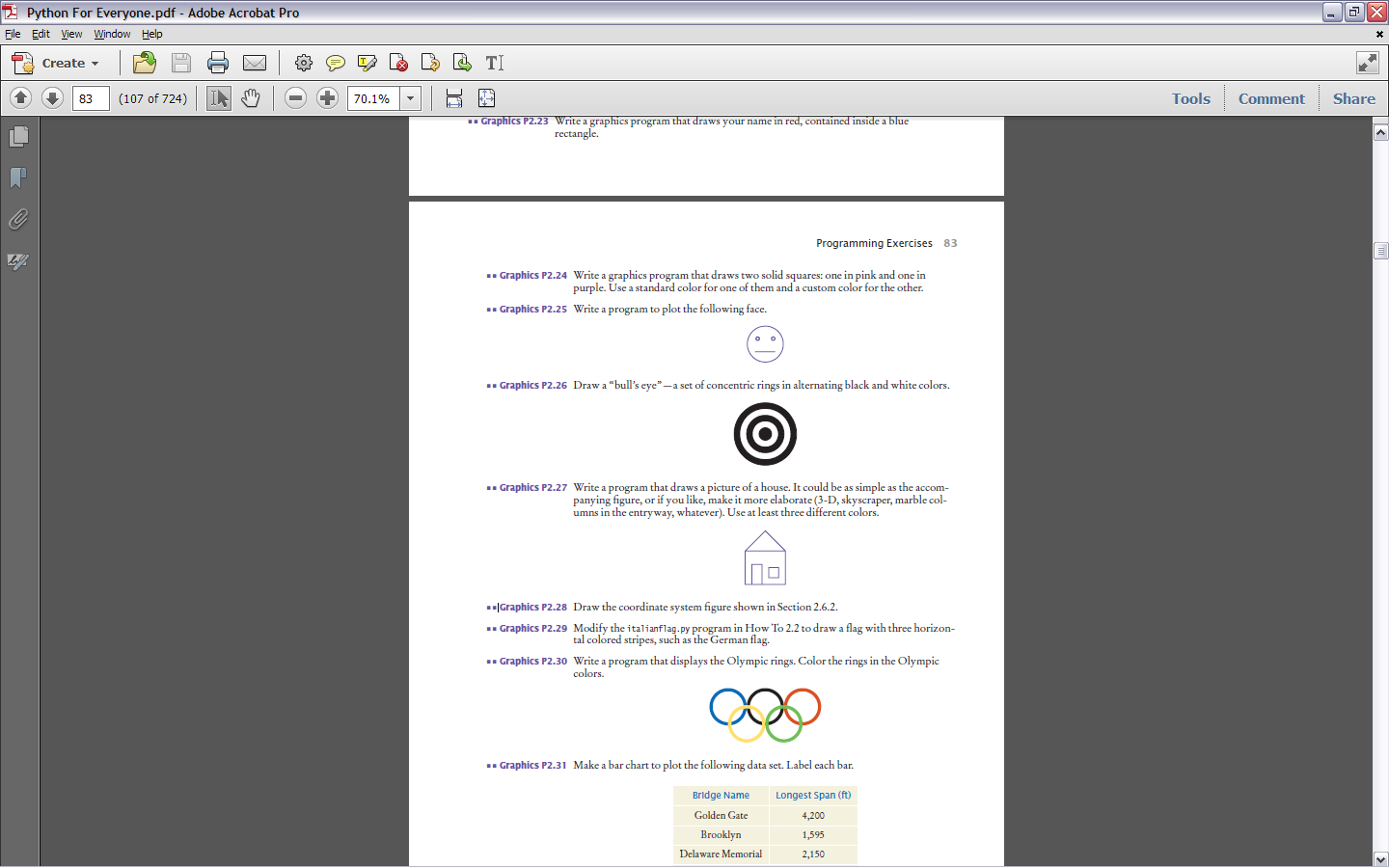
1. Write a graphics program that draws two solid squares: one in Red and one in

Green. Use a standard color for one of them and a custom color for the other.

1. Write a program to plot the following face.



1. Draw a “bull’s eye”—a set of concentric rings in alternating black and white colors.



1. Write a program that draws a picture of a house. It could be as simple as the accompanying figure.

