

2.6 Graphics

SIMPLE DRAWINGS

Drawing Simple Graphics

- To help you create simple drawings, we have included a graphics module with the book that is a simplified version of Python's more complex library module.
- The module code and usage instructions are available with the source code for the book on its companion web site.

Using the graphics module (1)

- To create a graphical application using the graphics module, carry out the following at the top of your program:

```
from graphics import GraphicsWindow
```

- Create a graphics window (640 x 480 pixels):

```
win = GraphicsWindow(640, 480)
```

- Access the canvas contained in the graphics window:

```
canvas = win.canvas()
```

Using the graphics module (2)

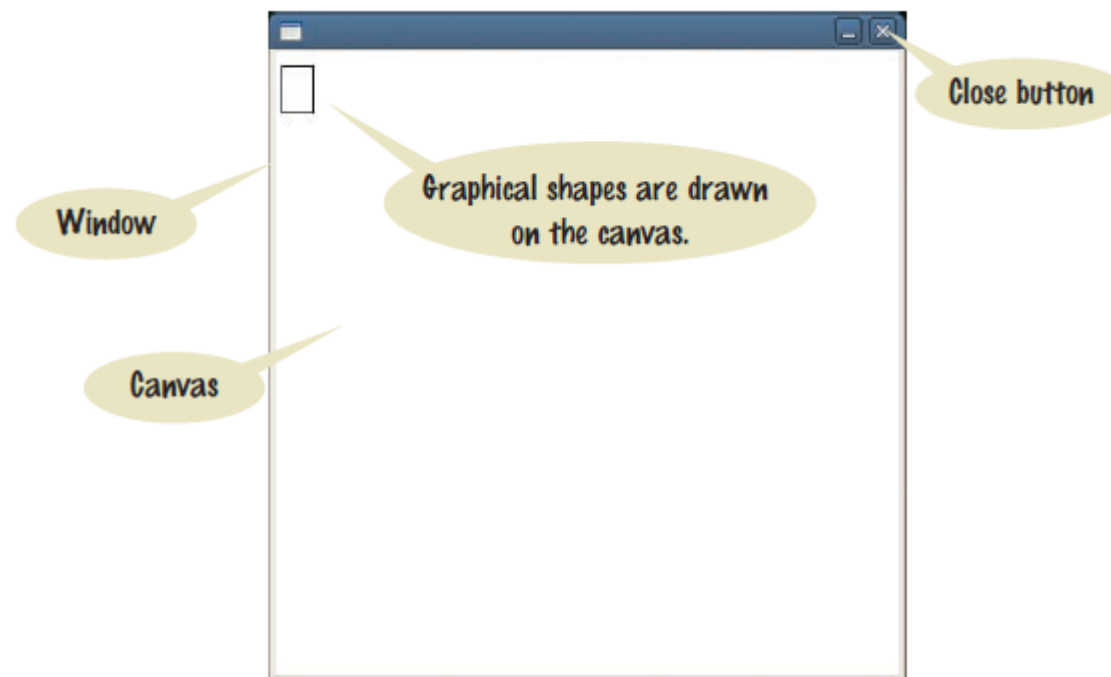
- Create your drawing.

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

- Have the program 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.

```
win.wait()
```

A graphics window



A complete drawing example

ch02/window.py

```
1  ##
2  # This program creates a graphics window with a rectangle. It provides the
3  # template used with all of the graphical programs used in the book.
4  #
5
6  from graphics import GraphicsWindow
7
8  # Create the window and access the canvas.
9  win = GraphicsWindow()
10 canvas = win.canvas()
11
12 # Draw on the canvas.
13 canvas.drawRect(5, 10, 20, 30)
14
15 # Wait for the user to close the window.
16 win.wait()
```

Table 10: GraphicsWindow Methods

Table 10 GraphicsWindow Methods	
Method	Description
$w = \text{GraphicsWindow}()$ $w = \text{GraphicsWindow}(width, height)$	Creates a new graphics window with an empty canvas. The size of the canvas is 400×400 unless another size is specified.
$w.\text{canvas}()$	Returns the object representing the canvas contained in the graphics window.
$w.\text{wait}()$	Keeps the graphics window open and waits for the user to click the “close” button.

Drawing shapes

- Basic shapes have 4 properties: **x coordinate**, **y coordinate**, **width** and **height**.

- Example:

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

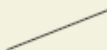

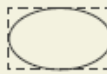

- This statement draws a rectangle with the upper top left corner at point (x = 15, y = 10) in the window with a height of 20 and a width of 30.
- Common shapes that can be drawn include: rectangles, squares, circles and ovals.

Drawing lines

- Lines require 4 slightly different properties than drawing shapes:
 - Point 1(x coordinate, y coordinate)
 - Point 2(x coordinate, y coordinate)

Table 13: Common Shapes, Lines and Text

Table 13 GraphicsCanvas Drawing Methods

Method	Result	Notes
<code>c.drawLine(x₁, y₁, x₂, y₂)</code>		(x_1, y_1) and (x_2, y_2) are the endpoints.
<code>c.drawRect(x, y, width, height)</code>		(x, y) is the top left corner.
<code>c.drawOval(x, y, width, height)</code>		(x, y) is the top-left corner of the box that bounds the ellipse. To draw a circle, use the same value for <i>width</i> and <i>height</i> .
<code>c.drawText(x, y, text)</code>		(x, y) is the anchor point.

The canvas and shapes can be colored

- If you use the default setting (not changing the fill or outline), shapes are outlined in black and there is no fill color.
- The fill color and outline can be set to different colors with the method calls:

```
setFill(<color name>)
```

OR

```
setFill(<red level>, <green level>, <blue level>)
```

```
setOutline(<color name>)
```

OR

```
setOutline(<red level>, <green level>, <blue level>)
```

Example of setting color

- The following statements draw a rectangle that is outlined in black and filled with green.

```
canvas.setOutline("black")
```

```
canvas.setFill(0, 255, 0)
```

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



Table 11: Common Color Names

Table 11 Common Color Names			
Color Name	Color Name	Color Name	Color Name
"black"	"magenta"	"maroon"	"pink"
"blue"	"yellow"	"dark blue"	"orange"
"red"	"white"	"dark red"	"sea green"
"green"	"gray"	"dark green"	"light gray"
"cyan"	"gold"	"dark cyan"	"tan"

Table 12: GraphicsCanvas Color Methods

Table 12 GraphicsCanvas Color Methods	
Method	Description
<code>c.setColor(<i>name</i>)</code> <code>c.setColor(<i>red</i>, <i>green</i>, <i>blue</i>)</code>	Sets both the fill and outline color to the same color. Color can be set by the color's <i>name</i> or by values for its <i>red</i> , <i>green</i> , and <i>blue</i> components.
<code>c.setFill()</code> <code>c.setFill(<i>name</i>)</code> <code>c.setFill(<i>red</i>, <i>green</i>, <i>blue</i>)</code>	Sets the color used to fill a geometric shape. If no argument is given, the fill color is cleared.
<code>c.setOutline()</code> <code>c.setOutline(<i>name</i>)</code> <code>c.setOutline(<i>red</i>, <i>green</i>, <i>blue</i>)</code>	Sets the color used to draw lines and text. If no argument is given, the outline color is cleared.

Summary: variables

- A variable is a storage location with a name.
- When defining a variable, you must specify an initial value.
- By convention, variable names should start with a lower case letter.
- An assignment statement stores a new value in a variable, replacing the previously stored value.

Summary: operators

- The assignment operator = does not denote mathematical equality.
- Variables whose initial value should not change are typically capitalized by convention.
- The / operator performs a division yielding a value that may have a fractional value.
- The // operator performs a division, the remainder is discarded.
- The % operator computes the remainder of a floor division.

Summary: python overview

- The Python library declares many mathematical functions, such as `sqrt()` and `abs()`
- You can convert between integers, floats and strings using the respective functions: `int()`, `float()`, `str()`
- Python libraries are grouped into modules. Use the `import` statement to use methods from a module.
- Use the `input()` function to read keyboard input in a console window.

Summary: python overview

- Use the format specifiers to specify how values should be formatted.

Summary: Strings

- Strings are sequences of characters.
- The `len()` function yields the number of characters in a String.
- Use the `+` operator to concatenate Strings; that is, to put them together to yield a longer String.
- In order to perform a concatenation, the `+` operator requires both arguments to be strings. Numbers must be converted to strings using the `str()` function.
- String index numbers are counted starting with 0.

Summary: Strings

- Use the [] operator to extract the elements of a String.

Summary: graphics

- Graphical shapes (such as squares, rectangles, circles, ovals), or lines and text can be drawn using the graphics module.
- The color of graphical objects can be set with the `setOutline()` and `setFill()` methods.

Chapter Goals

- To declare and initialize variables and constants
- To understand the properties and limitations of integers and floating-point numbers
- To appreciate the importance of comments and good code layout
- To write arithmetic expressions and assignment statements
- To create programs that read, and process inputs, and display the results
- To learn how to use Python strings
- To create simple graphics programs using basic shapes and text