1 Administrative Topics

• Dale is teaching while Stephanie is away. These are the notes she would have used.

• A note about conditional statements. Python will let you write:

\[
x == 'hi' \text{ or } 'bye'
\]

but it won’t be doing what you want it to do, which is likely this:

\[
x == 'hi' \text{ or } x == 'bye'
\]

2 Objects vs Modules

Today we are going to learn about more of the objects available in the graphics package written by John Zelle. But first I want to reiterate the difference between modules and objects.

• Module
  
  – What it is: a collection of functions
  – How you get one: import
  – How many do you have: Each file will have only one copy of each module
• Object
  – What it is: a collection of data and methods that operate on that data
  – How you get one: create one using a function. For an object that will be a Circle, you create it using the function called Circle
  – How many do you have: Each code file can create as many objects as it needs

3 Graphics Objects

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window</td>
<td>an object that allows you to put graphics into a window</td>
</tr>
<tr>
<td>Point</td>
<td>an object that allows you to specify the x-y coordinates of a graphics object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shape Objects Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>a circular graphics object – lets you draw a circle</td>
</tr>
<tr>
<td>Rectangle</td>
<td>a rectangular graphics object – lets you draw a rectangle</td>
</tr>
<tr>
<td>Oval</td>
<td>an elliptic graphics object – lets you draw an oval</td>
</tr>
<tr>
<td>Line</td>
<td>a line graphics object</td>
</tr>
<tr>
<td>Polygon</td>
<td>a graphics object that is constructed by providing all the vertices</td>
</tr>
</tbody>
</table>

The Graphwin methods we are interested in are:

• win.getMouse() – waits for a mouse click
• win.close() – closes the window
• win.checkMouse() – True if a mouse has recently clicked on the window

Zelle has organized the methods for the shape functions well. There are some methods (and functions) specific to each shape. Most notably, the functions used to create each object are unique:

• Circle( centerpt, radius ). centerpt is a Point specifying the location of the center of the circle. radius is a number.
• **Rectangle( pt1, pt2 ).** each pt is a Point specifying opposite corners of the rectangle (e.g. upper left and lower right).

• **Oval( pt1, pt2 ).** Each pt is a Point specifying opposite corners of the rectangle that bounds the oval you want to draw.

• **Line( pt1, pt2 ).** Each pt is an endpoint of the line segment.

• **Polygon( pt1, pt2, pt3, ... ) or Polygon([ pt1, pt2, pt3, ... ] ).** Each pt is a vertex of a polygon.

There are several methods that are general - they can be used on any of the shape objects. Suppose that *obj* is a Circle, Rectangle, Oval, Line, or Polygon. Then, the follow methods are defined:

• *obj.draw(win)* – places *obj* in the window (*win*) (which must be a `GraphWin`)

• *obj.move(dx, dy)* – moves the *obj* by a given amount

• *obj.setOutline(clr)* – sets the outline color

• *obj.setWidth(width)* – sets the width of the outline (in pixels)

• *obj.setFill(clr)* – sets the fill color

For the color-setting methods, *clr* must be either a string with a color name (e.g. ‘green’, ‘blue’, etc.) or the output of the function `color_rgb` (e.g. `graphics.color_rgb(255, 150, 20)`).

To draw a simple circle, you use code like this:

```python
# Make a window big enough to hold the scene
w = 300
h = 300
win = graphics.GraphWin("Crazy Toy", w, h)

# Make a circle centered at 150,150 with radius 20
center = graphics.Point(150, 150)
c = graphics.Circle(center, 20)
```
# Draw the circle (i.e. put it in the window)
c.draw(win)

# Keep the window open for awhile
win.getMouse()  # Pause to view result
win.close()     # Close window when done
The color/width-setting methods that allow us to do make more interesting scenes.

Below is a picture of a very simplified view of a Fisher Price Person.

```python
# Make a window big enough to hold the scene
w = 300
h = 300
win = graphics.GraphWin("Crazy Toy", w, h)

# Make a person
upper_left = graphics.Point(w/2-20, 150)
lower_right = graphics.Point(w/2+20, 150+60)
body = graphics.Rectangle(upper_left, lower_right)
body.setFill('blue')
body.setOutline('blue')
center = graphics.Point(w/2, 150-20)
head = graphics.Circle(center, 20)
head.setOutline(graphics.color_rgb(0,0,0))
head.setFill(graphics.color_rgb(255,0,255))

# Draw the objects.
body.draw(win)
head.draw(win)

# Keep the window open for awhile
win.getMouse()  # Pause to view result
win.close()    # Close window when done
```

Let’s think about the symbol table contents after this code has run. Suppose we have executed all the lines except the last line. What does the main symbol table look like?

There are 6 graphics objects (1 GraphWin, 3 Points, 1 Rectangle, and 1 Circle).
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>300 &lt;int&gt;</td>
</tr>
<tr>
<td>h</td>
<td>300 &lt;int&gt;</td>
</tr>
<tr>
<td>win</td>
<td>→ &lt;GraphWin&gt;</td>
</tr>
<tr>
<td>upper_left</td>
<td>→ &lt;Point&gt;</td>
</tr>
<tr>
<td>lower_right</td>
<td>→ &lt;Point&gt;</td>
</tr>
<tr>
<td>body</td>
<td>→ &lt;Rectangle&gt;</td>
</tr>
<tr>
<td>center</td>
<td>→ &lt;Point&gt;</td>
</tr>
<tr>
<td>head</td>
<td>→ &lt;Circle&gt;</td>
</tr>
</tbody>
</table>