

Graphics

15-110 – Bonus Content

How Tkinter Works – Starter Code

In order to create graphics, we need to take a few preliminary steps. These will be provided to you as starter code, inside `makeCanvas()`.

First, **create a new window**- that's the thing that pops up on the screen.

Second, **create a new canvas**- that's the thing we can draw graphics on.

Next, **pack the canvas into the window**- that tells the canvas to fill the whole window.

We'll do all our drawing in `draw()`.

Finally, the last line will tell the window to stay open until we press the X button.

```
import tkinter as tk # shorten library name
def draw(canvas):
    pass
```

```
def makeCanvas(w, h):
    root = tk.Tk()
    canvas = tk.Canvas(root, width=w, height=h)
    canvas.configure(bd=0,
                    highlightthickness=0)
```

```
canvas.pack()
```

```
draw(canvas)
```

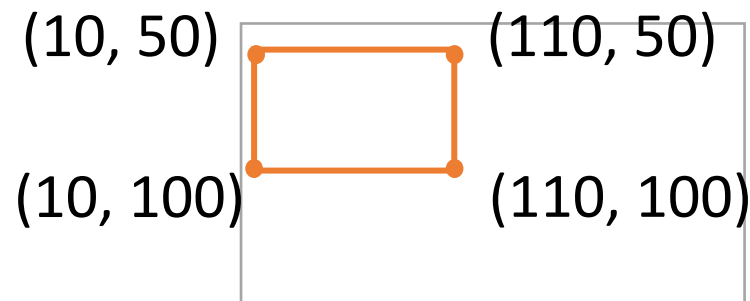
```
root.mainloop()
```

```
makeCanvas(400, 400)
```

Drawing a Rectangle

To draw a rectangle, we use the function `create_rectangle` in our `draw` function. This function takes four required parameters: the x and y coordinates of the **left-top** corner, and the x and y coordinates of the **right-bottom** corner. The rectangle will then be drawn between those two points.

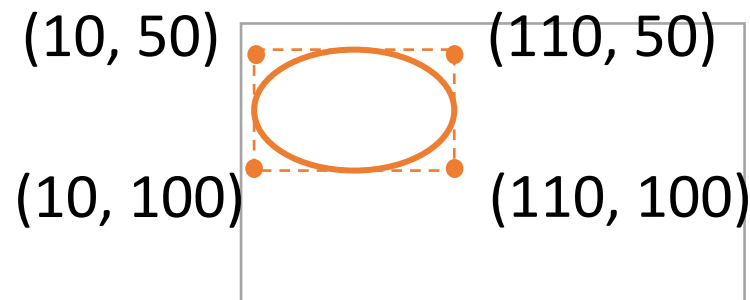
```
canvas.create_rectangle(10, 50, 110, 100)
```



Drawing an Oval

We can draw more shapes than just rectangles. To draw an oval, use `create_oval`. This function uses the same parameters as `create_rectangle`, where the coordinates mark the oval's **bounding box**.

```
canvas.create_oval(10, 50, 110, 100)
```



Drawing Squares/Circles

If you want to draw a square or a circle, you need to ensure that the width of the shape equals the height.

How can you do that? Make sure that (right - left) is equal to (bottom - top)!

```
canvas.create_rectangle(50, 100, 150, 200)
```

Keyword Arguments Add Variety

With the basic parameters, we can only draw basic outlines of shapes. By adding **keyword arguments**, we can change the properties of these shapes.

Each keyword argument is associated with a **default value**, which is why we don't need to include them in every graphics call. To change that default value, include the keyword, followed by `=`, followed by the new value in the function call.

```
canvas.create_rectangle(50, 100, 150, 200, fill="green")
```

Keyword Argument - fill

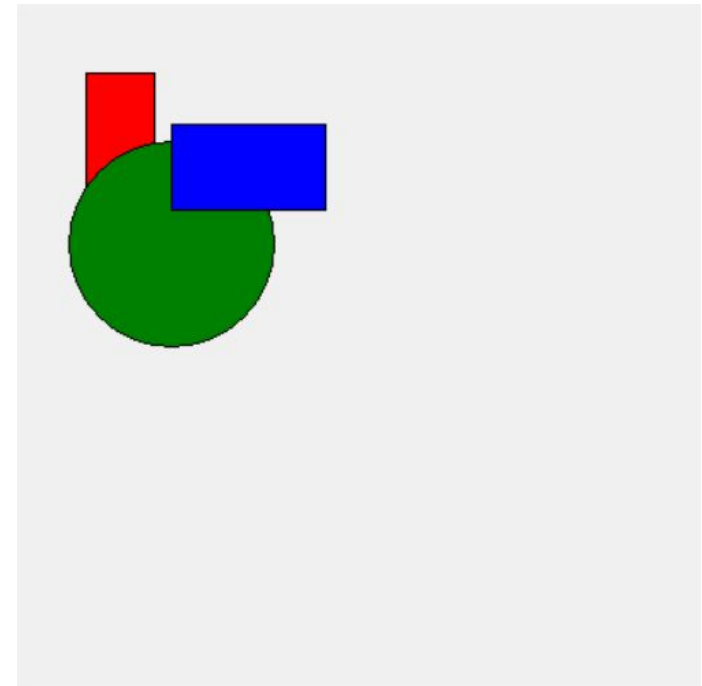
The `fill` argument can be used on any shape. It uses a string (the name of the color) to change the color of the shape.

```
canvas.create_rectangle(40, 40, 80, 140, fill="red")
```

```
canvas.create_oval(30, 80, 150, 200, fill="green")
```

```
canvas.create_rectangle(90, 70, 180, 120, fill="blue")
```

Note that when we draw shapes on top of each other, the one on top is the **last one called**. Order matters!



Sidebar: Hexadecimal Numbers For Colors

What if we want to define our own colors using the RGB system we discussed in the Data Representation system? Python lets us do this, but we'll need to represent the RGB values in a new number system.

We use the **hexadecimal** number system to represent a byte with just two digits. This number system uses base **16**. In comparison, normal decimal uses base **10** and binary uses base **2**.

The digits of hexadecimal are : 0123456789**ABCDEF**

Example: 01111011 = 7B, because 0111 is 7 and 1011 is 11 (B).

Making New Colors

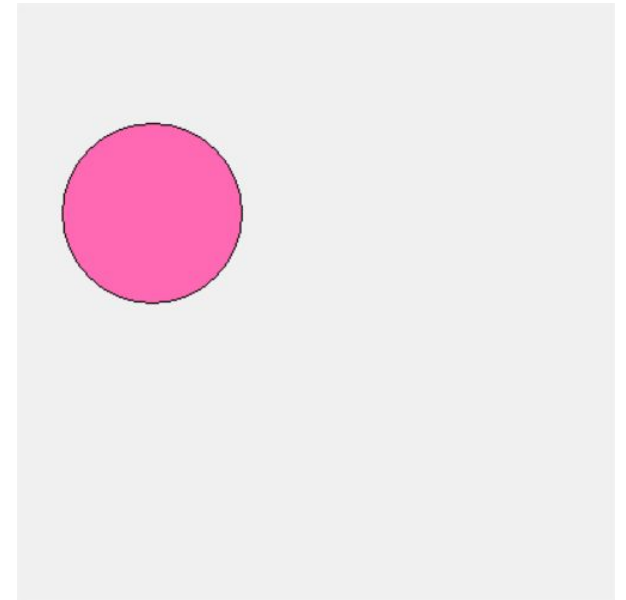
To define a new color, make a string "#RRGGBB", where you replace RR with the red value in hex, GG with green, and BB with blue.

"#FF69B4" is hot pink!

```
canvas.create_oval(30, 80, 150, 200, fill="#FF69B4")
```

Interested in finding more Tkinter color names? There's a whole databank!

<https://wiki.tcl-lang.org/page/Color+Names%2C+running%2C+all+screens>



Keyword Argument - width

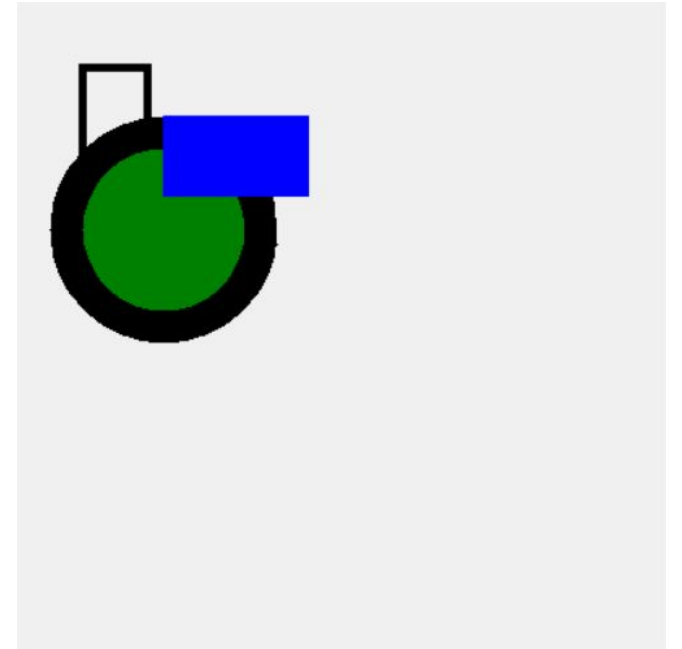
Another keyword argument is `width`, which specifies how many pixels wide the border of the shape should be.

```
canvas.create_rectangle(40, 40, 80, 140, width=5)
```

```
canvas.create_oval(30, 80, 150, 200,  
                  width=20, fill="green")
```

```
canvas.create_rectangle(90, 70, 180, 120,  
                        fill="blue", width=0)
```

Note that setting `width` to `0` removes the border completely.



Drawing Lines

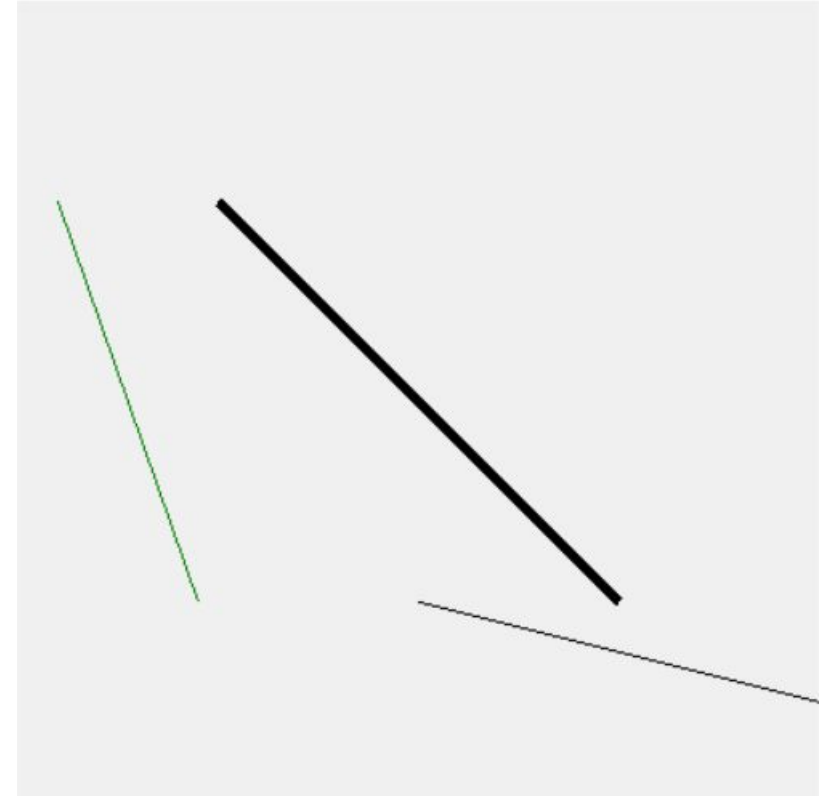
To draw a line on the screen, you specify the two endpoints of the line.

```
canvas.create_line(200, 300, 400, 350)
```

```
canvas.create_line(20, 100, 90, 300, fill="green")
```

```
canvas.create_line(100, 100, 300, 300, width=5)
```

Again, we can use **fill** and **width** to modify the lines.



Drawing Text

Drawing text on the canvas works a bit differently from drawing rectangles, ovals, and lines. We specify only one coordinate – the pixel where the **center** of text will be drawn.

```
canvas.create_text(200, 200, text="Hello World")
```

Although `text` is keyword argument and technically optional, `text` is required in order to draw text at all.

Keyword Argument - font

When drawing text, we can use the keyword argument `font` to change the appearance of the text.

The font parameters takes a string with one to three pieces of information – the font name, the font size, and the font type.

```
canvas.create_text(200, 200, text="Hello World!",  
                  font="Arial")
```

```
canvas.create_text(100, 100, text="This is fun!",  
                  font="Times 30")
```

```
canvas.create_text(300, 300, text="weewooweewoo",  
                  font="Courier 10 italic")
```

You can find a full list of fonts and types here:

<https://effbot.org/tkinterbook/tkinter-widget-styling.htm#fonts>



Keyword Argument - anchor

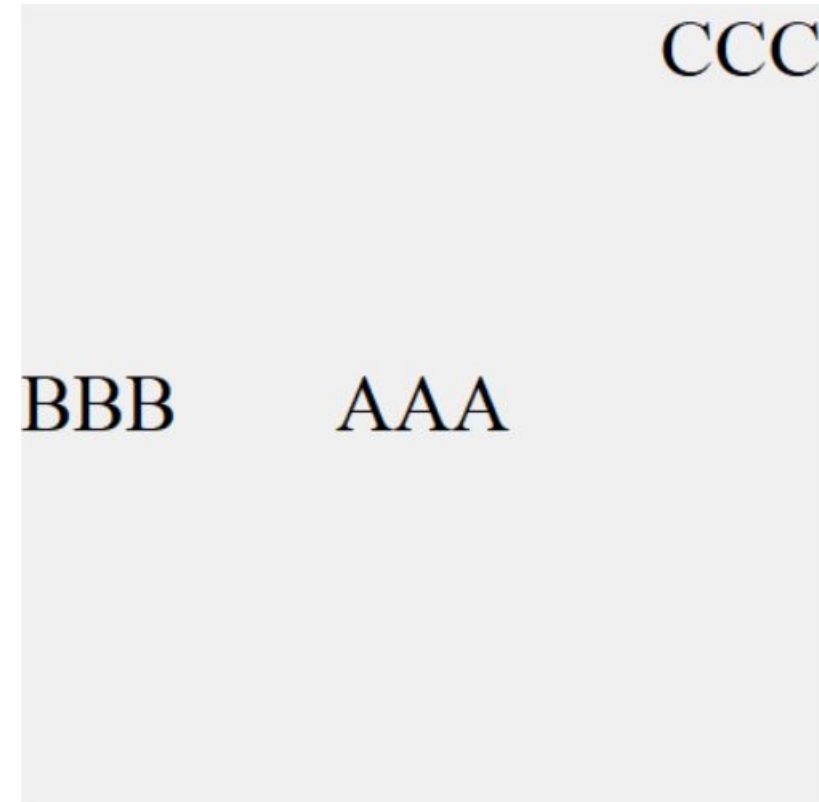
The point used in the `canvas.create_text` call is actually an **anchor** for the text, to describe where it is drawn from. That anchor defaults to the center of the text box, but we can change it to be any compass point instead.

```
canvas.create_text(200, 200, text="AAA",  
                  font="Times 30", anchor="center")
```

```
canvas.create_text(0, 200, text="BBB",  
                  font="Times 30", anchor="w")
```

```
canvas.create_text(400, 0, text="CCC",  
                  font="Times 30", anchor="ne")
```

Note that the anchor describes the **point on the text box** that will correspond to the (x, y) coordinate. Since CCC's anchor is "ne" (north-east), the upper-right corner of the text box is placed at (400, 0).



Tkinter Can Do Even More!

There's plenty of things Tkinter can draw and plenty of additional keyword arguments that we haven't covered here.

If you're interested in learning more, check out the Tkinter documentation: <http://effbot.org/tkinterbook/canvas.htm>