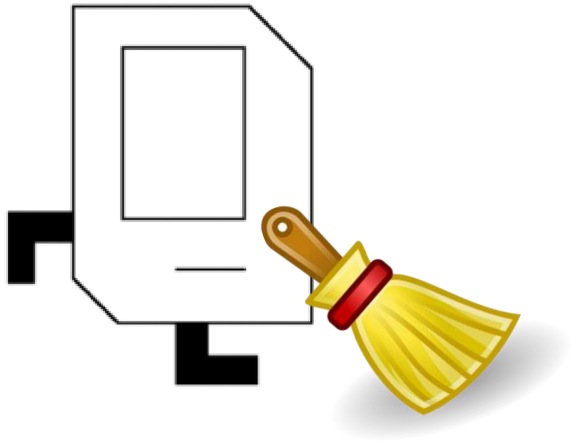
Graphics

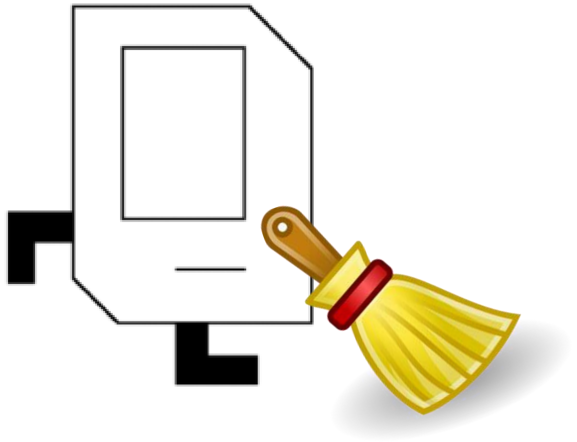
CS106A, Stanford University

Housekeeping I



- Handout #10: Graphics Reference Guide
  - We'll talk about graphics today
- Assignment #3 due today
  - Pain poll: http://PollEv.com/mehransahami943
- Assignment #4 released today
  - Due May 9th (almost a week after midterm)
  - Sandcastle problems on lists of lists and strings
    - Do those to get practice on those topics before the midterm

Housekeeping II

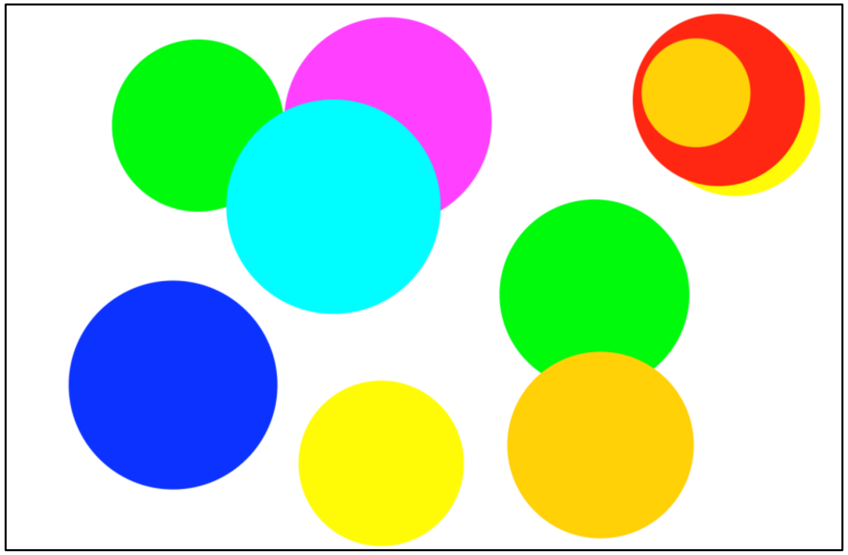


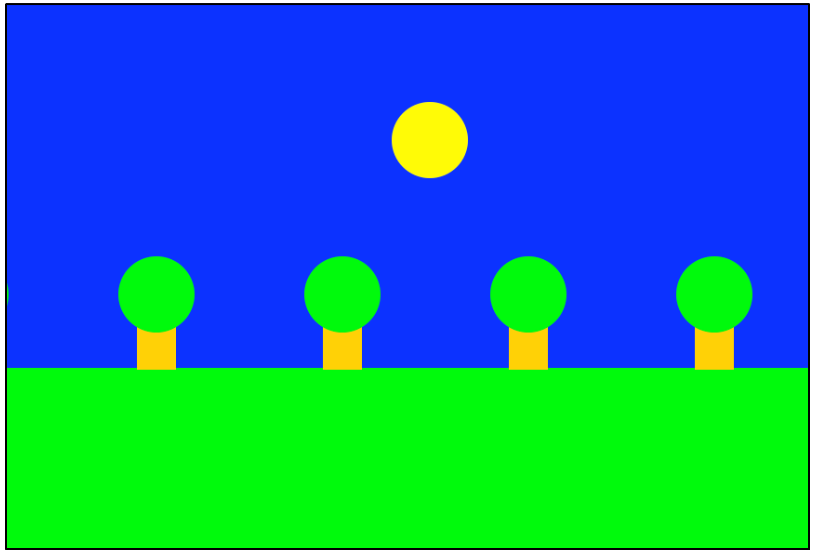
- Midterm will be on Tuesday, May 3rd from 7pm-9pm
  - If you have an unmovable academic conflict, please let me know via email by TODAY (April 25th) at 5pm
  - In your email, please list ***all*** the times you'd be available to take an alternate midterm between May 2nd and 4th
  - I'll notify you by April 30th of alternate midterm time
- We will provide a practice midterm later this week, so you can get a sense of topics on actual exam

Today's Goals

1. Learning about drawing basic graphics in Python
2. Creating programs that draw pictures

Graphics Programs





Graphics with `tkinter`

- We want to draw pictures in Python
- Use a simple graphics library called **`tkinter`**
  - You need to import this library at the top of your program

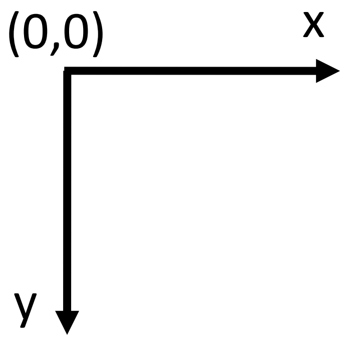
```
import tkinter
```

- Then you create a canvas to draw on
  - We'll provide code that creates the canvas (looks like this):

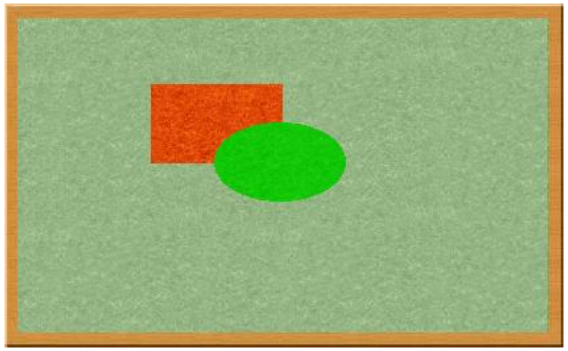
```
import tkinter

CANVAS_WIDTH = 600        # Width of canvas in pixels
CANVAS_HEIGHT = 200       # Height of canvas in pixels

def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT)

    # drawing code called here (canvas passed as param)

    tkinter.mainloop()
```

Canvas

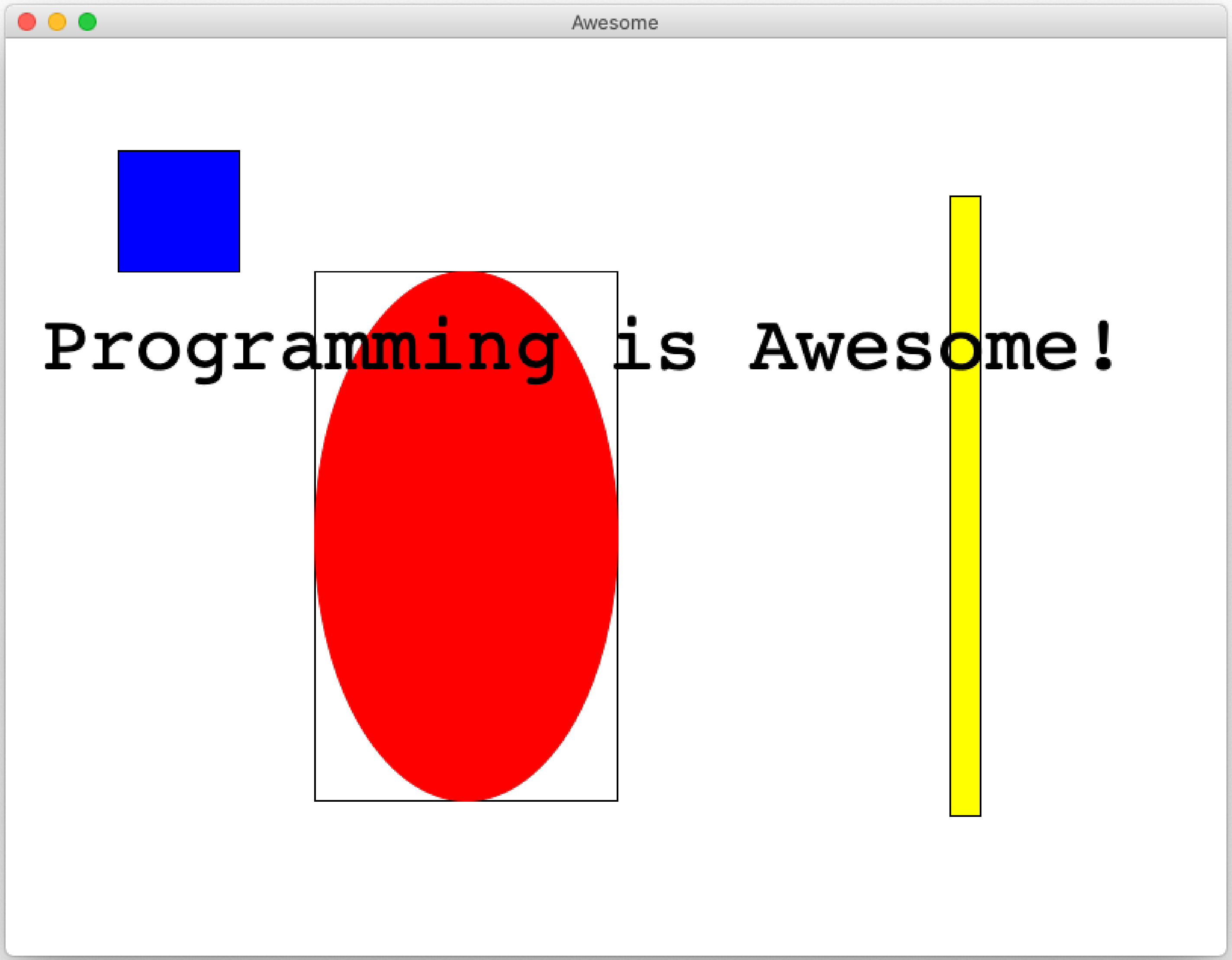
- The **canvas** is a where to make your drawings
  - The canvas is a grid of pixels
  - The origin (0, 0) is at the upper-left corner
  - y **<u>increases</u>** going down, x increases going right
  - Similar to an image, but canvas is not an image



- Drawing model is like a *collage* (or felt board)
  - You create shapes/text on the canvas
  - The shapes/text added to canvas have a *stacking order*
  - The objects we'll look at adding to a canvas include:
    - Rectangles
    - Ovals
    - Lines
    - Text

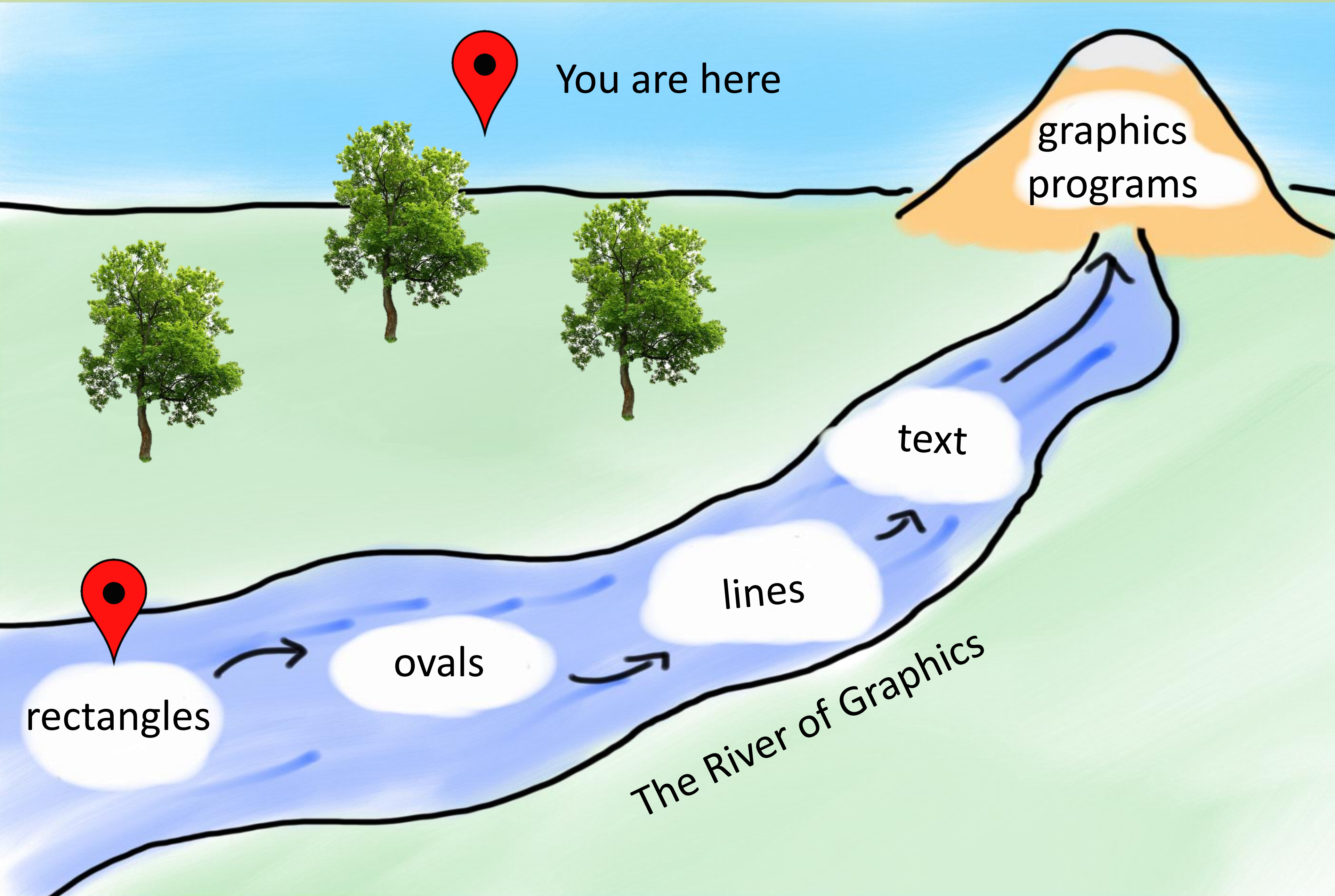


Rectangles, Ovals, Text



Programming is Awesome!

Today's Route



Creating Rectangles

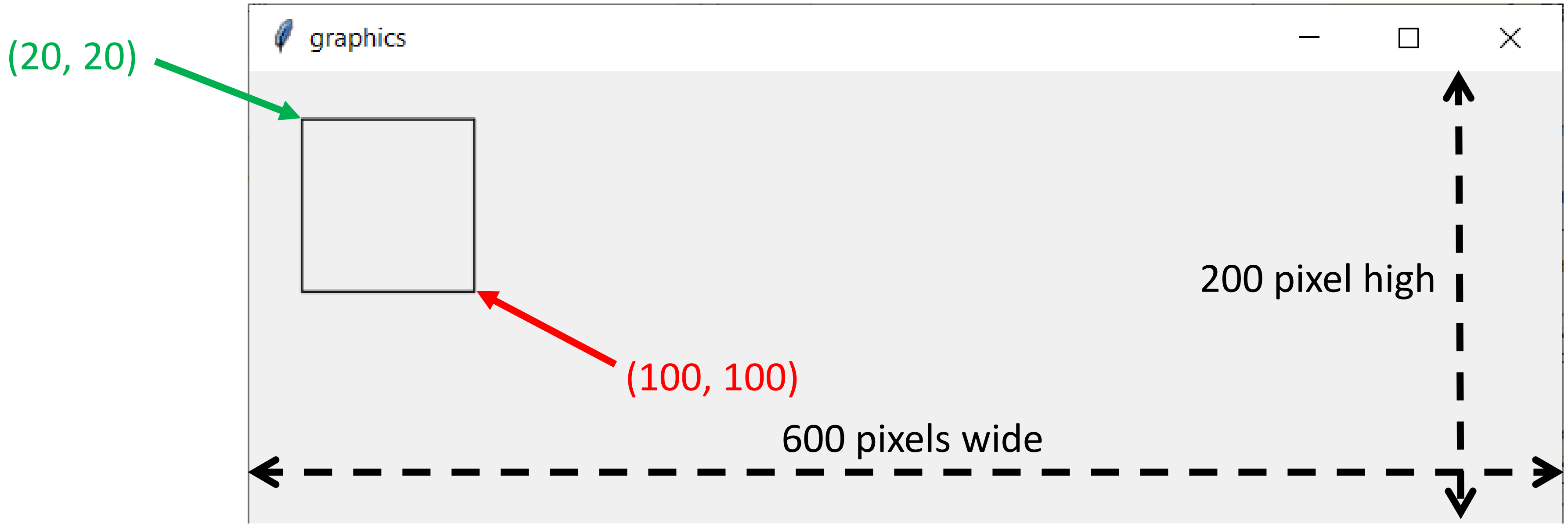
- Create a rectangle on a canvas
  - Call function **`create_rectangle`**
  - Specify upper left-hand corner (up_x, up_y) and lower right-hand corner (low_x, low_y) of the rectangle
- General form:

```
canvas.create_rectangle(up_x, up_y, low_x, low_y)
```

```
CANVAS_WIDTH = 600        # Width of canvas in pixels
CANVAS_HEIGHT = 200       # Height of canvas in pixels

def drawing(canvas):
    canvas.create_rectangle(20, 20, 100, 100)

def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT)
    drawing(canvas)
    tkinter.mainloop()
```

Creating Rectangles

```
CANVAS_WIDTH = 600        # Width of canvas in pixels
CANVAS_HEIGHT = 200       # Height of canvas in pixels

def drawing(canvas):
    canvas.create_rectangle(20, 20, 100, 100)

def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT)
    drawing(canvas)
    tkinter.mainloop()
```



Colored and Filled Rectangles

- Default rectangle is a black outline (no fill)
- Can specify color of rectangle outline with parameter named **`outline`**. For example:

```
canvas.create_rectangle(10, 10, 50, 50, outline='blue')
```

- Can specify a fill color for rectangle with parameter named **`fill`**. For example:

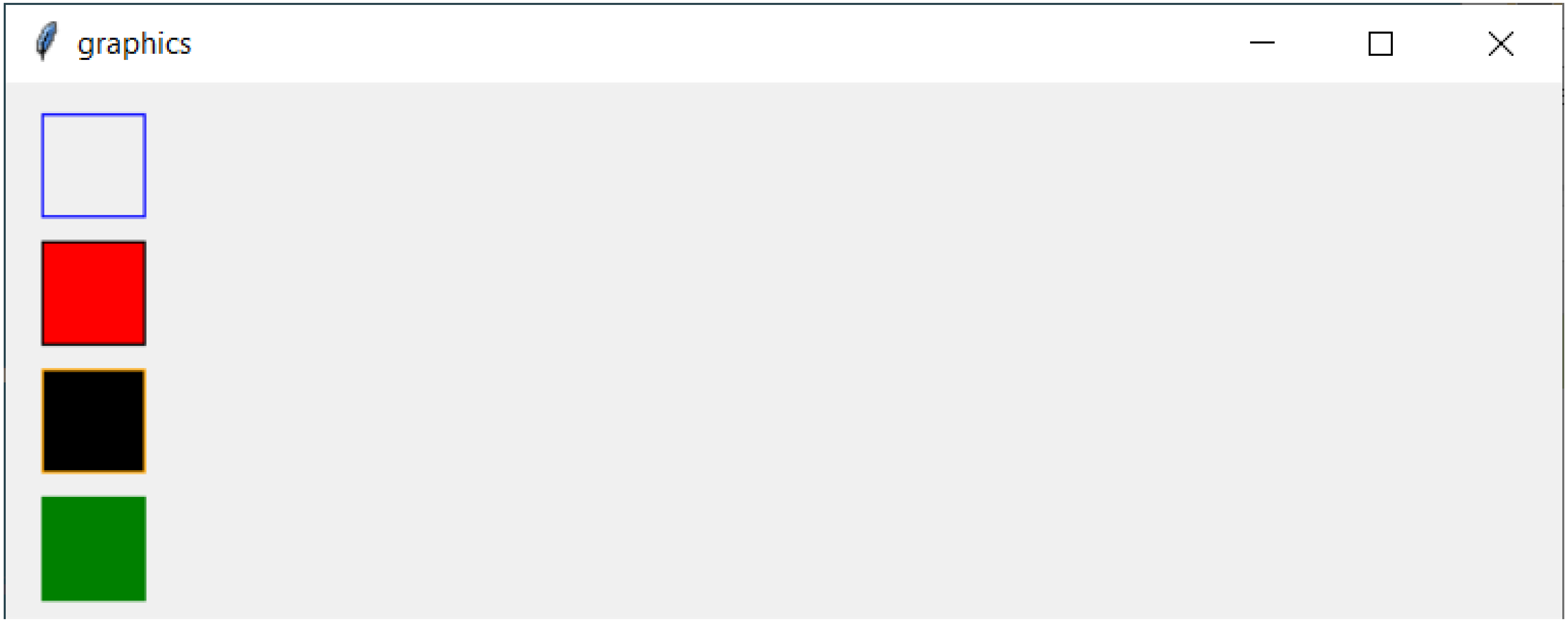
```
canvas.create_rectangle(10, 60, 50, 100, fill='red')
```

- Can also use both of these parameters together

```
def drawing(canvas):
    canvas.create_rectangle(10, 10, 50, 50, outline='blue')
    canvas.create_rectangle(10, 60, 50, 100, fill='red')
    canvas.create_rectangle(10, 110, 50, 150, fill='black',
        outline='orange')
    canvas.create_rectangle(10, 160, 50, 200, fill='green',
        outline='green')
```

Colored and Filled Rectangles

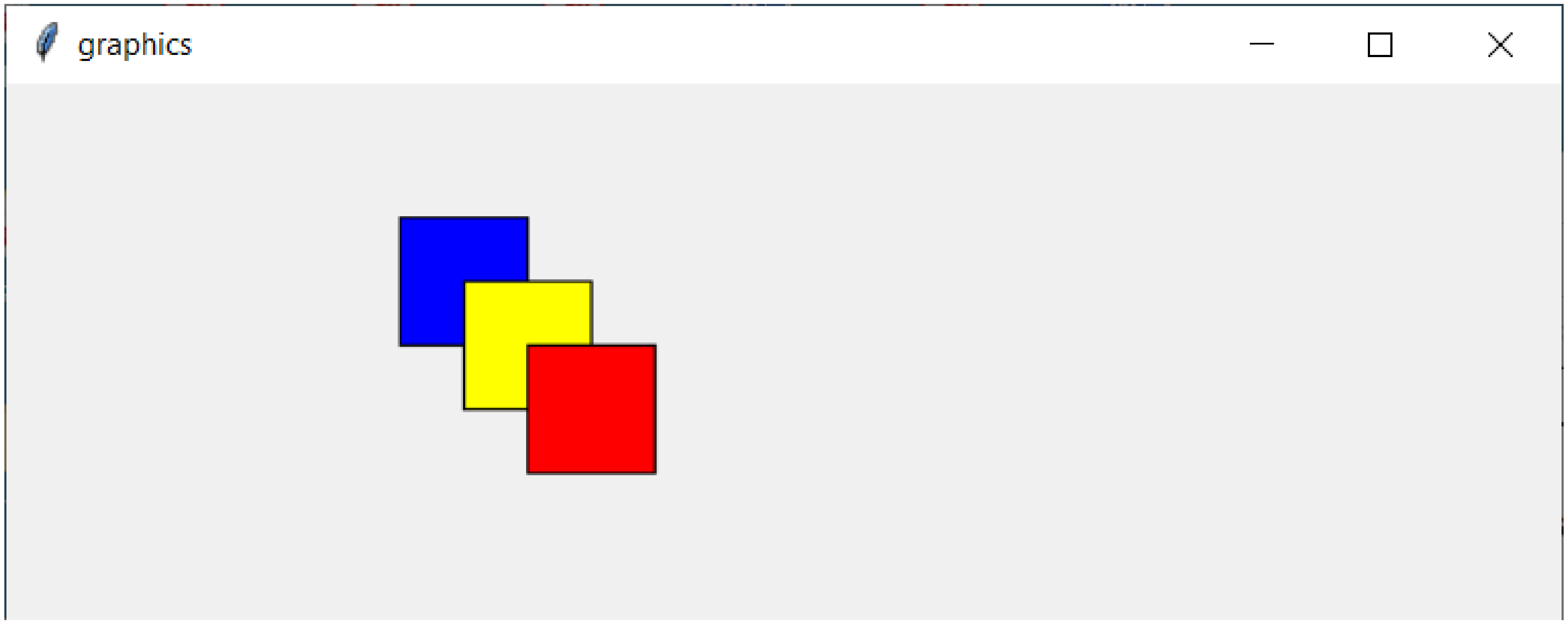
```
def drawing(canvas):
    canvas.create_rectangle(10, 10, 50, 50, outline='blue')
    canvas.create_rectangle(10, 60, 50, 100, fill='red')
    canvas.create_rectangle(10, 110, 50, 150, fill='black',
        outline='orange')
    canvas.create_rectangle(10, 160, 50, 200, fill='green',
        outline='green')
```



Stacking Order

- Note the order in which rectangles are drawn on the canvas

```
def drawing(canvas):
    canvas.create_rectangle(150, 50, 200, 100, fill='blue')
    canvas.create_rectangle(175, 75, 225, 125, fill='yellow')
    canvas.create_rectangle(200, 100, 250, 150, fill='red')
```



Colors

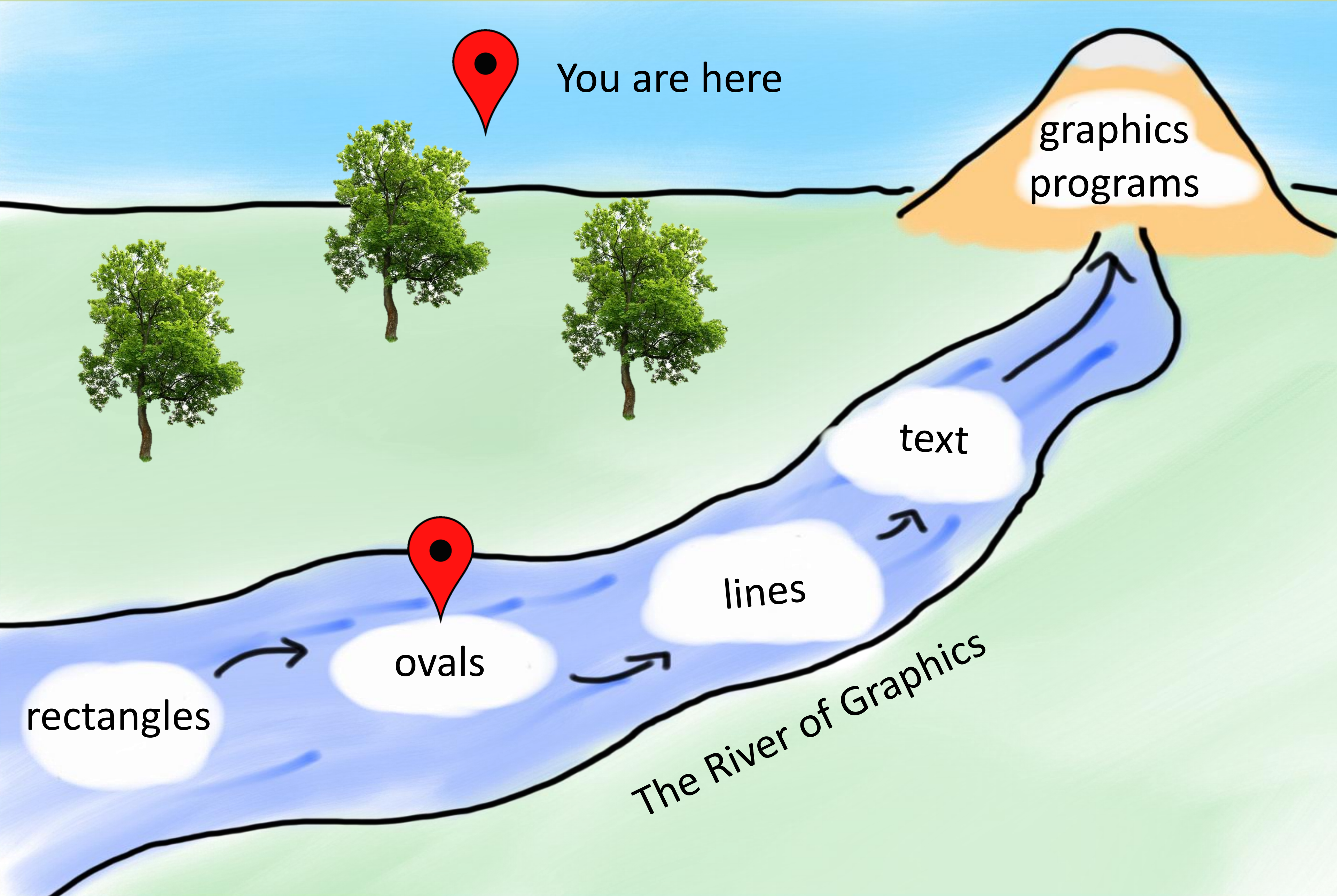
- `tkinter` has many built in colors. Here is a sample:

```
red          brown
blue         orange
green        gray
yellow       pink
white        tan
black        chartreuse
purple
```

- Can find the full (ridiculously long) list of colors at:

  https://www.tcl.tk/man/tcl8.6/TkCmd/colors.html

Today's Route

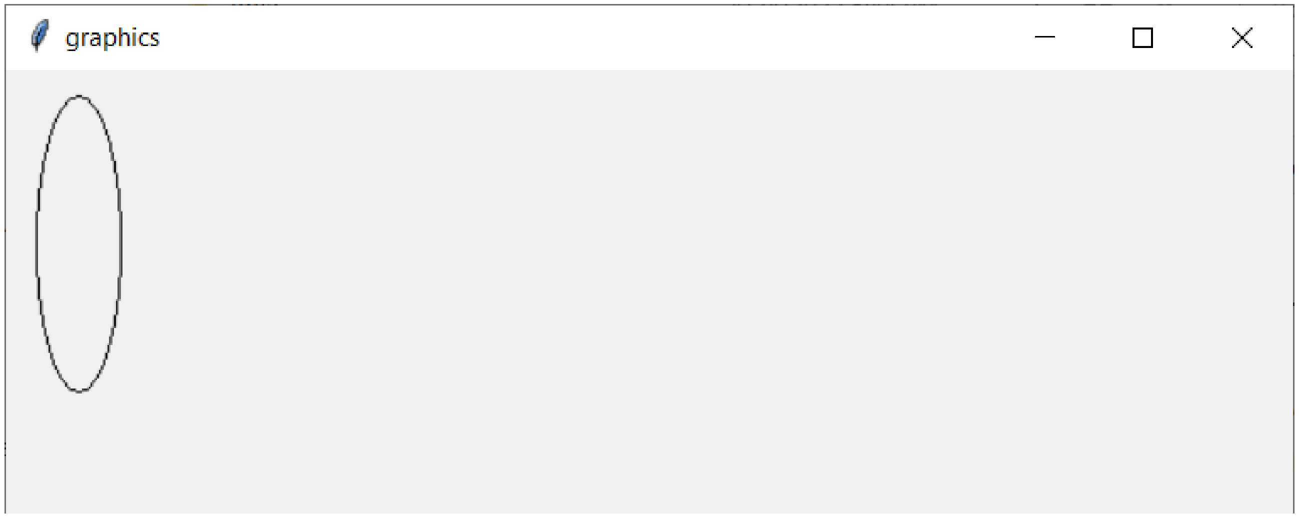


Creating Ovals

- Create an oval on a canvas
  - Call function **`create_oval`**
  - Specify upper left-hand corner (up_x, up_y) and lower right-hand corner (low_x, low_y) of the ***bounding box*** for oval
- General form:

```
canvas.create_oval(up_x, up_y, low_x, low_y)
```

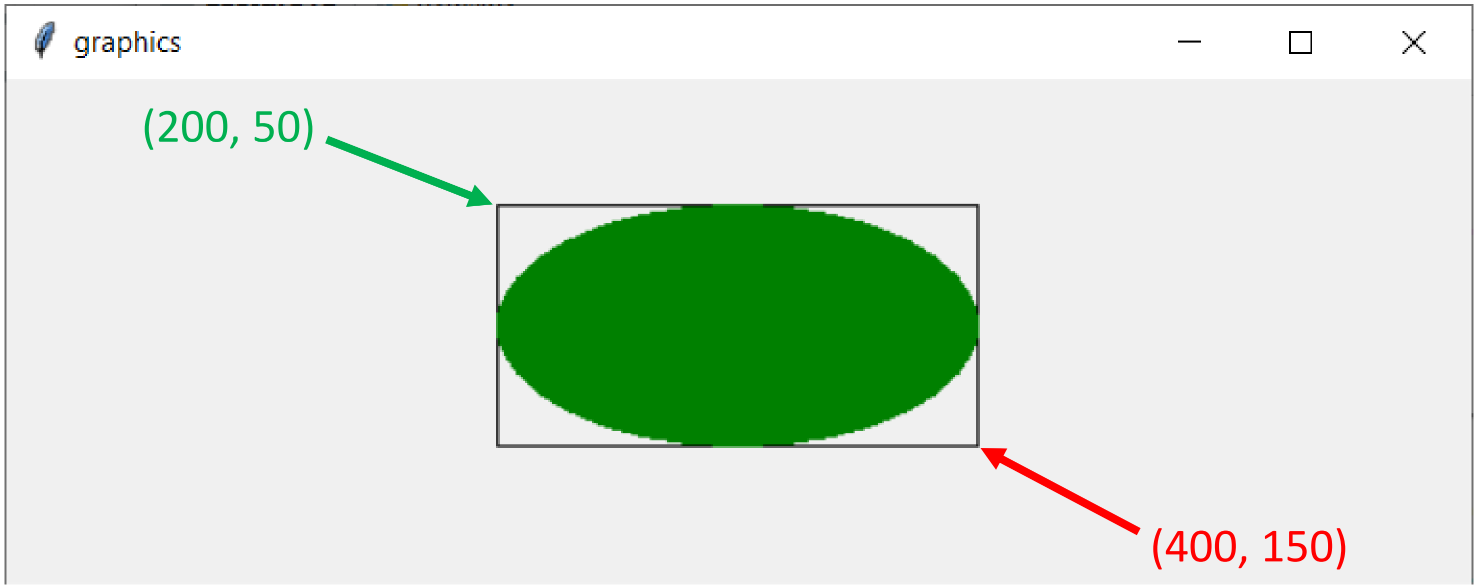
```
def drawing(canvas):
    canvas.create_oval(10, 10, 50, 150)
```



Understanding Bounding Box

- Oval is defined by bounding box:
  - Specify upper left-hand corner (up_x, up_y) and lower right-hand corner (low_x, low_y) of the *bounding box* for oval

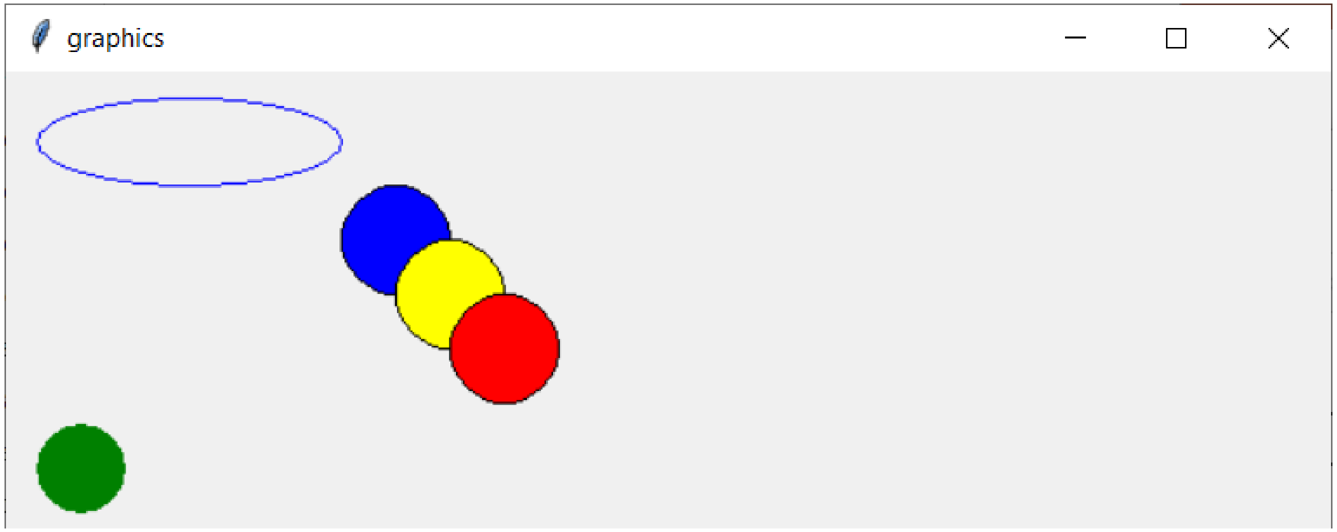
```
def drawing(canvas):
    # To show bounding box relative to a rectangle
    canvas.create_rectangle(200, 50, 400, 150)
    canvas.create_oval(200, 50, 400, 150,
        outline='green', fill='green')
```



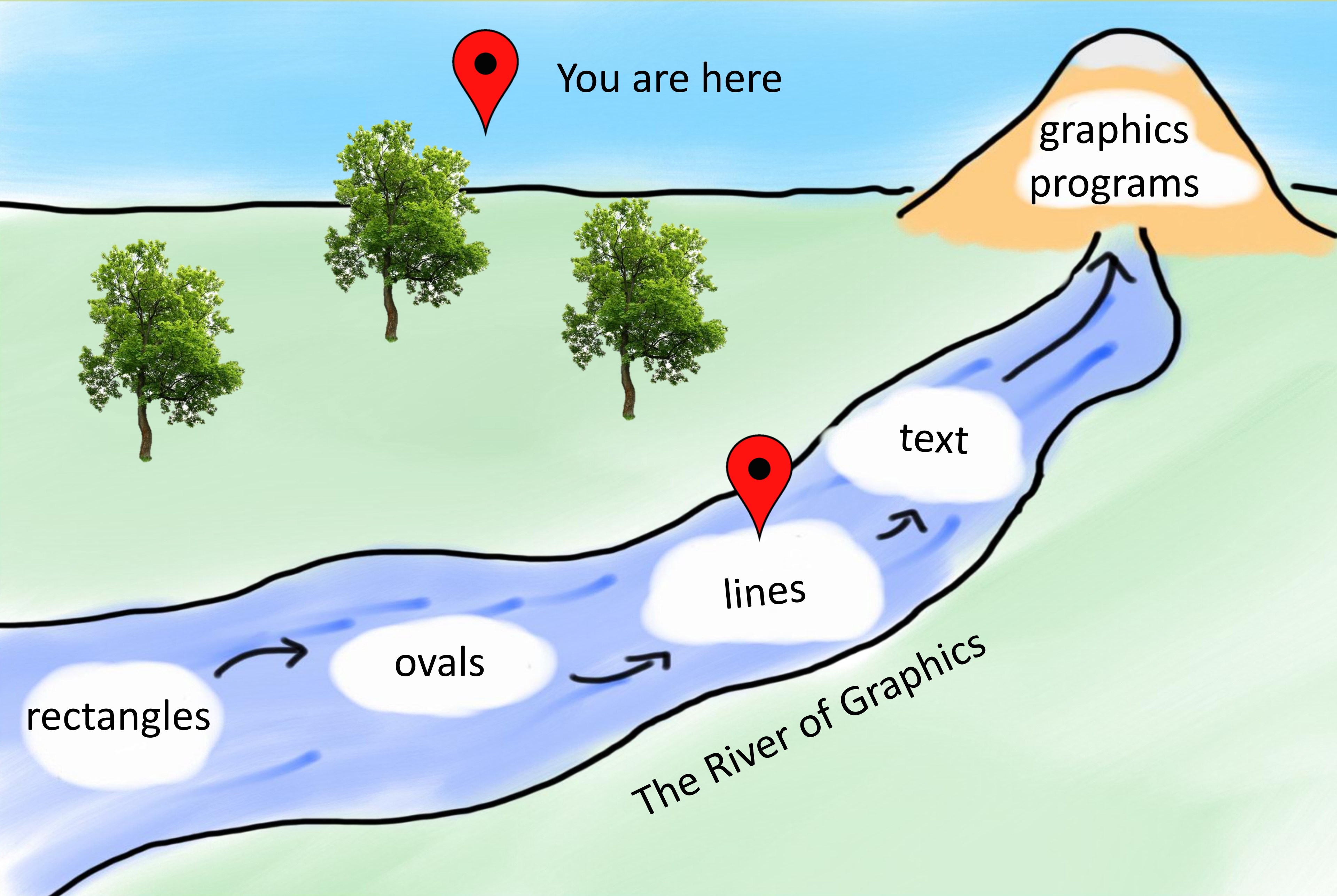
Colored and Filled Ovals

- Default oval is a black outline (no fill)
  - Can specify color of oval outline with parameter **`outline`**
  - Can specify a fill color for oval with parameter **`fill`**

```
def drawing(canvas):
    canvas.create_oval(10, 10, 150, 50, outline='blue')
    canvas.create_oval(10, 160, 50, 200, fill='green',
        outline='green')
    canvas.create_oval(150, 50, 200, 100, fill='blue')
    canvas.create_oval(175, 75, 225, 125, fill='yellow')
    canvas.create_oval(200, 100, 250, 150, fill='red')
```



Today's Route

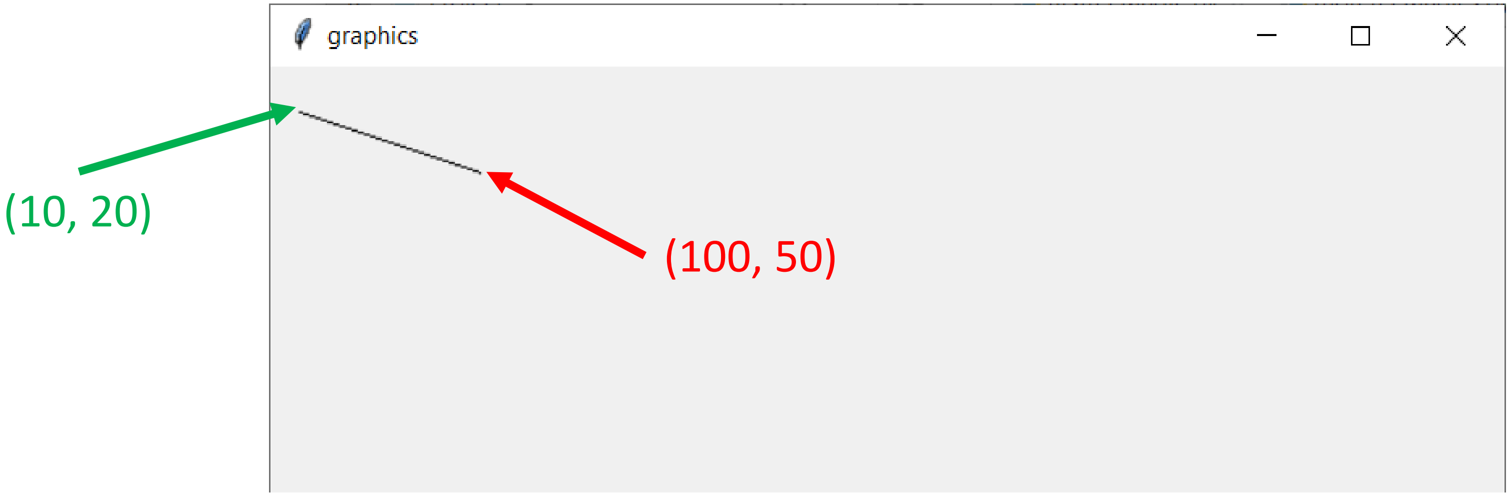


Creating Lines

- Create a line on a canvas
  - Call function **`create_line`**
  - Specify starting location (x1, y1) and ending location (x2, y2) of the line
- General form:

```
canvas.create_line(x1, y1, x2, y2)
```

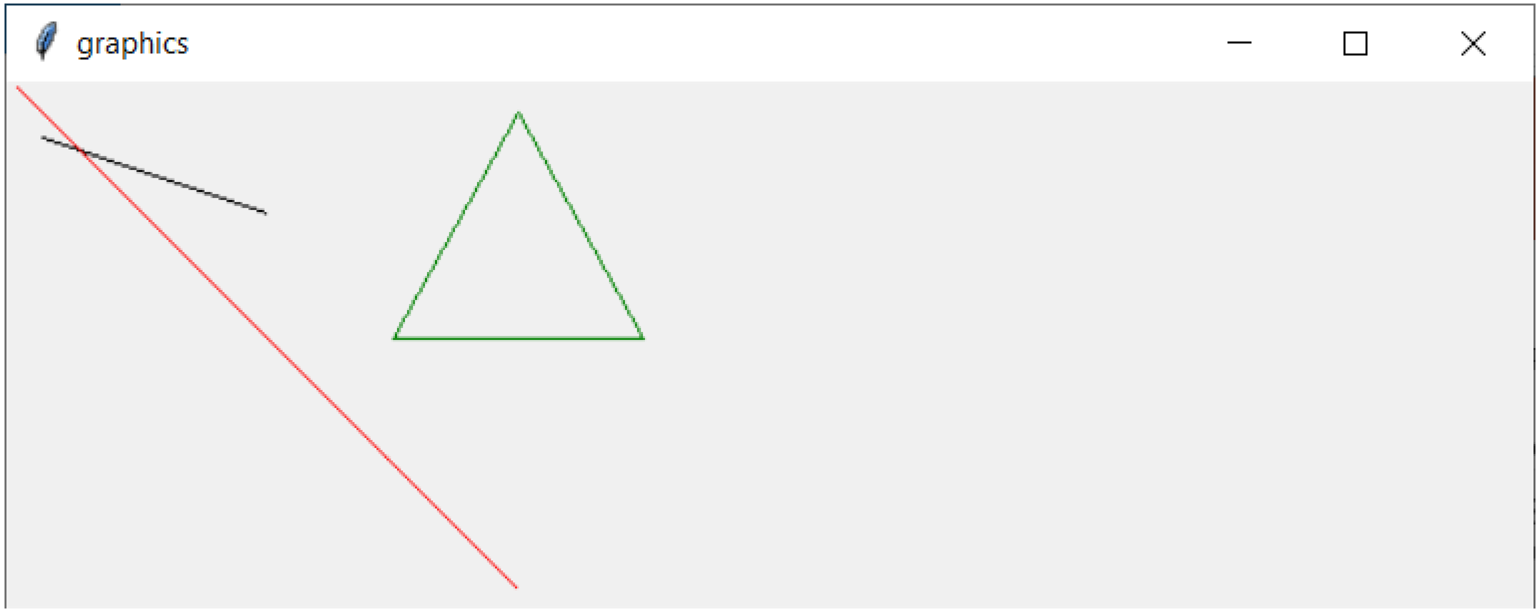
```
def drawing(canvas):
    canvas.create_line(10, 20, 100, 50)
```



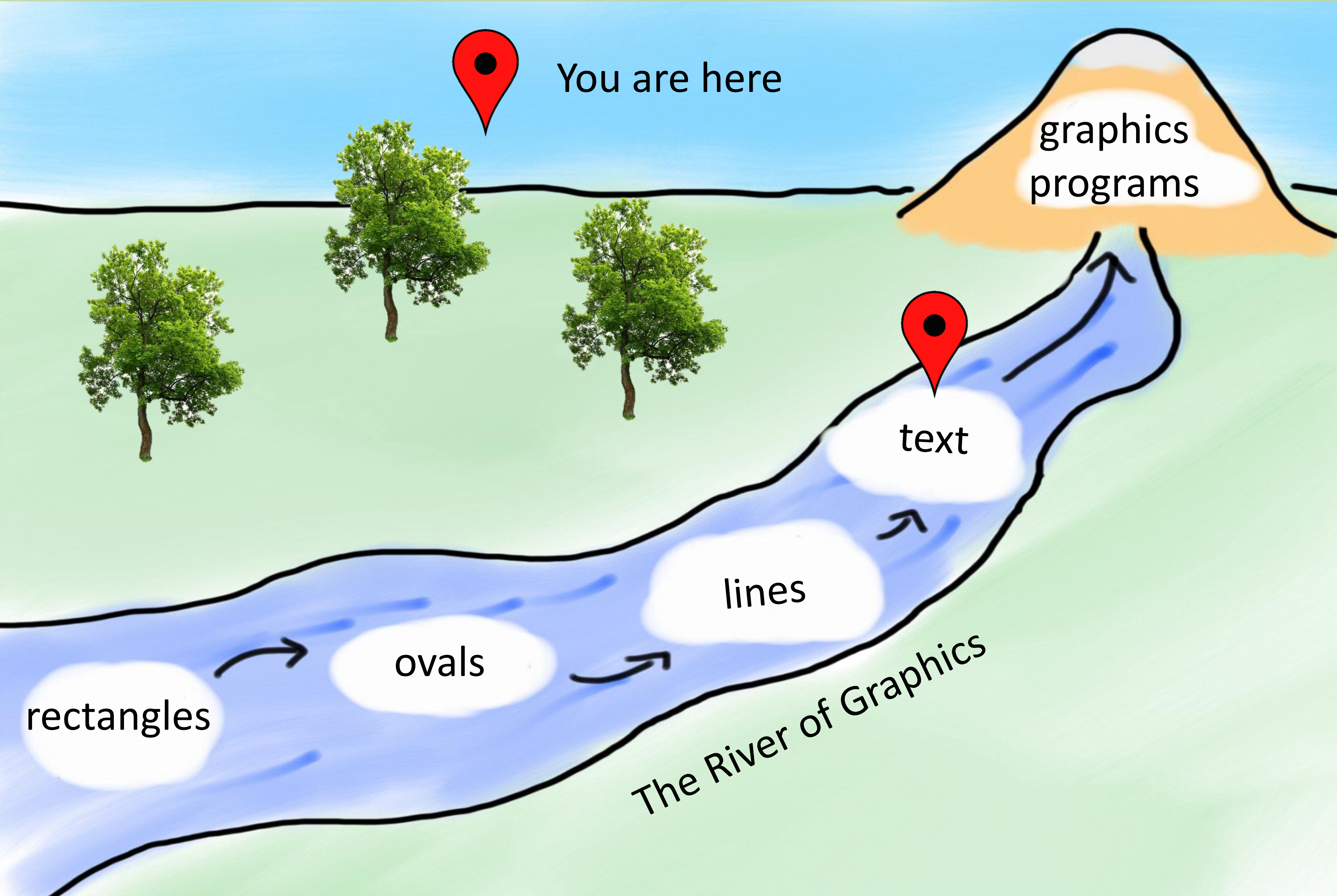
Colored Lines

- Default line is black
  - Can specify a color for line with parameter **`fill`**

```
def drawing(canvas):
    canvas.create_line(10, 20, 100, 50)
    canvas.create_line(0, 0, 200, 200, fill='red')
    canvas.create_line(200, 10, 150, 100, fill='green')
    canvas.create_line(150, 100, 250, 100, fill='green')
    canvas.create_line(250, 100, 200, 10, fill='green')
```



Today's Route

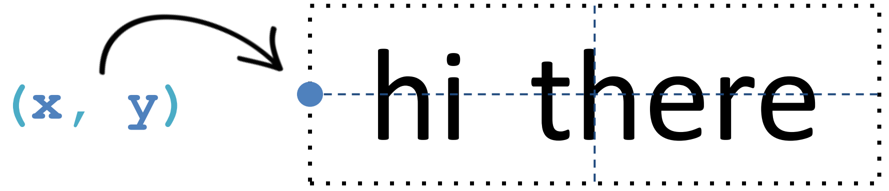


Creating Text

- Create text on a canvas
  - Call function **`create_text`**
  - Specify starting location (x, y) of the text, the anchor location, the font, and the actual text
  - For anchor, we use **`'w'`** for West, which means (x, y) location specifies starting point on the left-hand/West side of text
- General form:

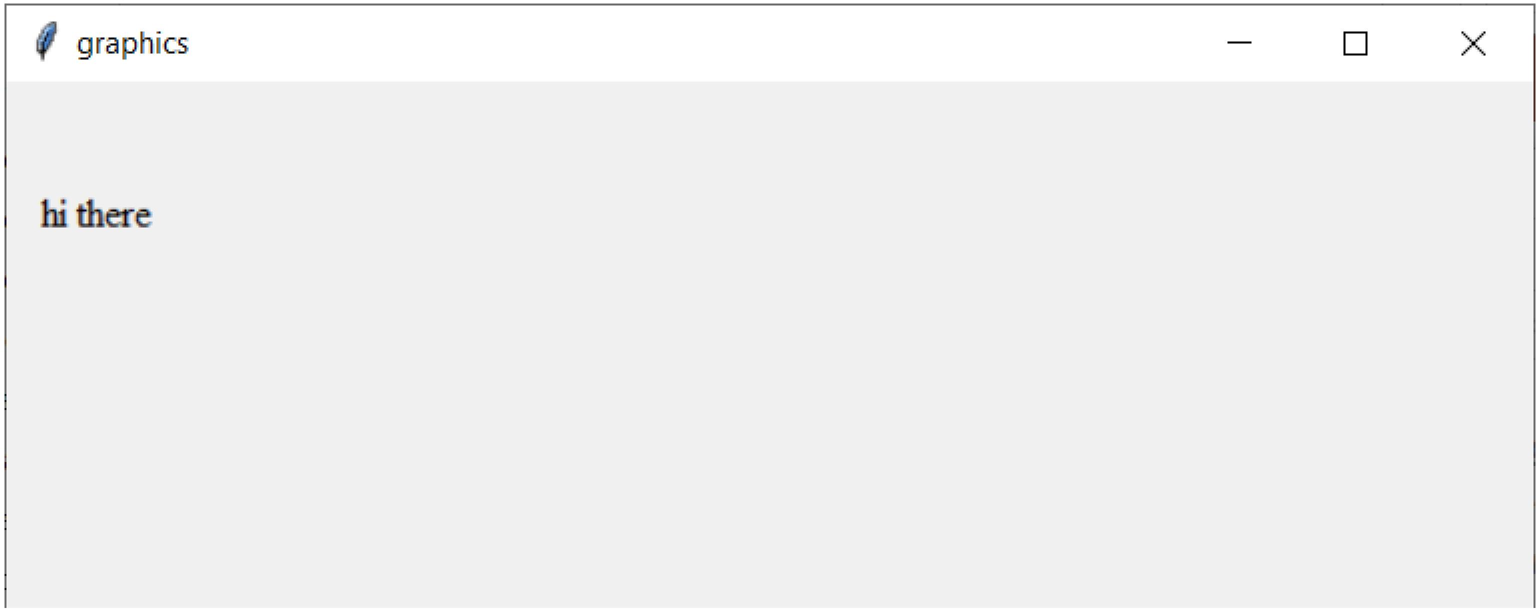
```
canvas.create_text(x, y, anchor='w', font='Times',
      text='text to display')
```

```
def drawing(canvas):
    canvas.create_text(10, 50, anchor='w', font='Times',
          text='hi there')
```



Creating Text

```
def drawing(canvas):
    canvas.create_text(10, 50, anchor='w', font='Times',
        text='hi there')
```

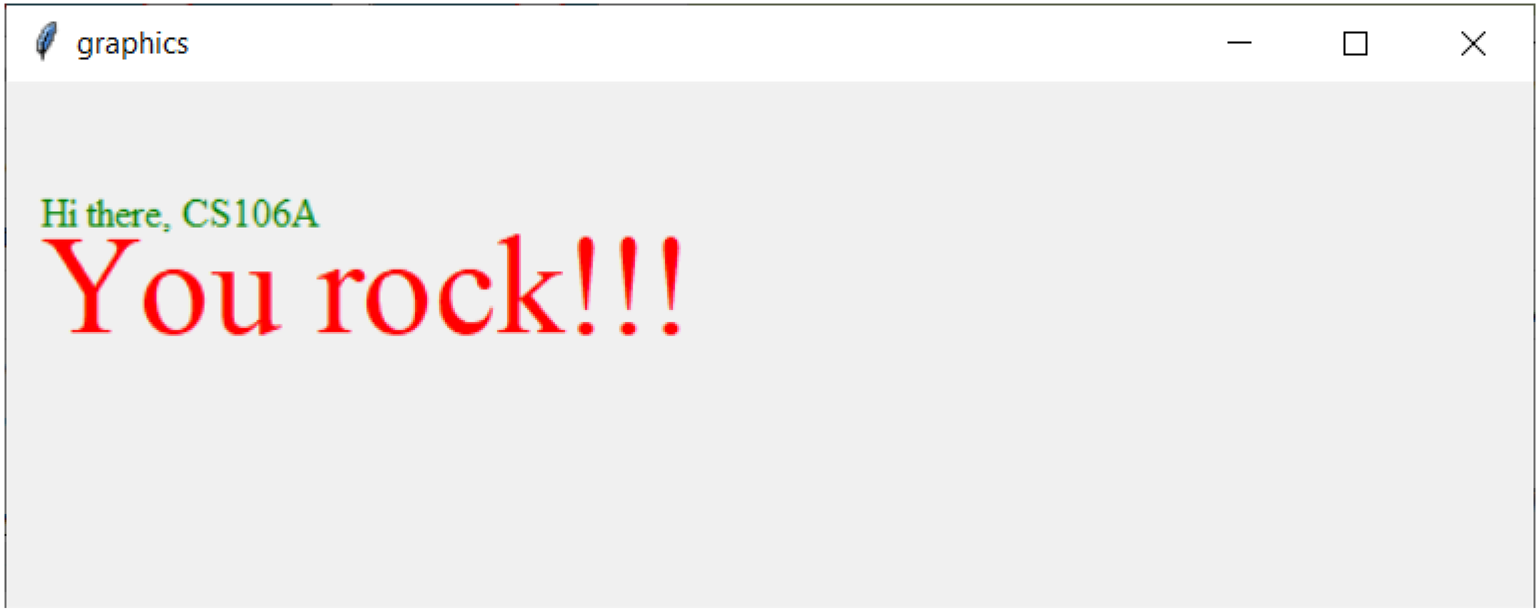


hi there

Can You Have Colored Text?!

- Default text is black
  - Can specify a color for text with parameter **`fill`**

```
def drawing(canvas):
    canvas.create_text(10, 50, anchor='w', font='Times',
                       text='Hi there, CS106A', fill='green')
    canvas.create_text(10, 80, anchor='w', font='Times 42',
                       text='You rock!!!', fill='red')
```

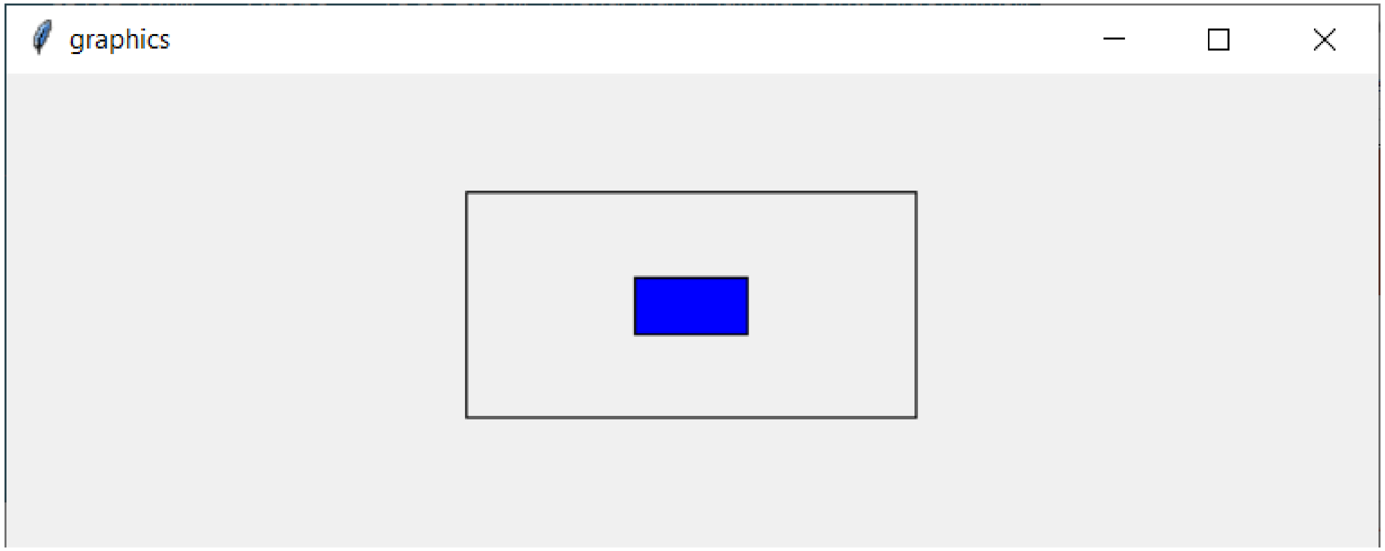


Hi there, CS106A
You rock!!!

Centering Objects in Drawings

- Say we want to draw rectangles centered on the canvas

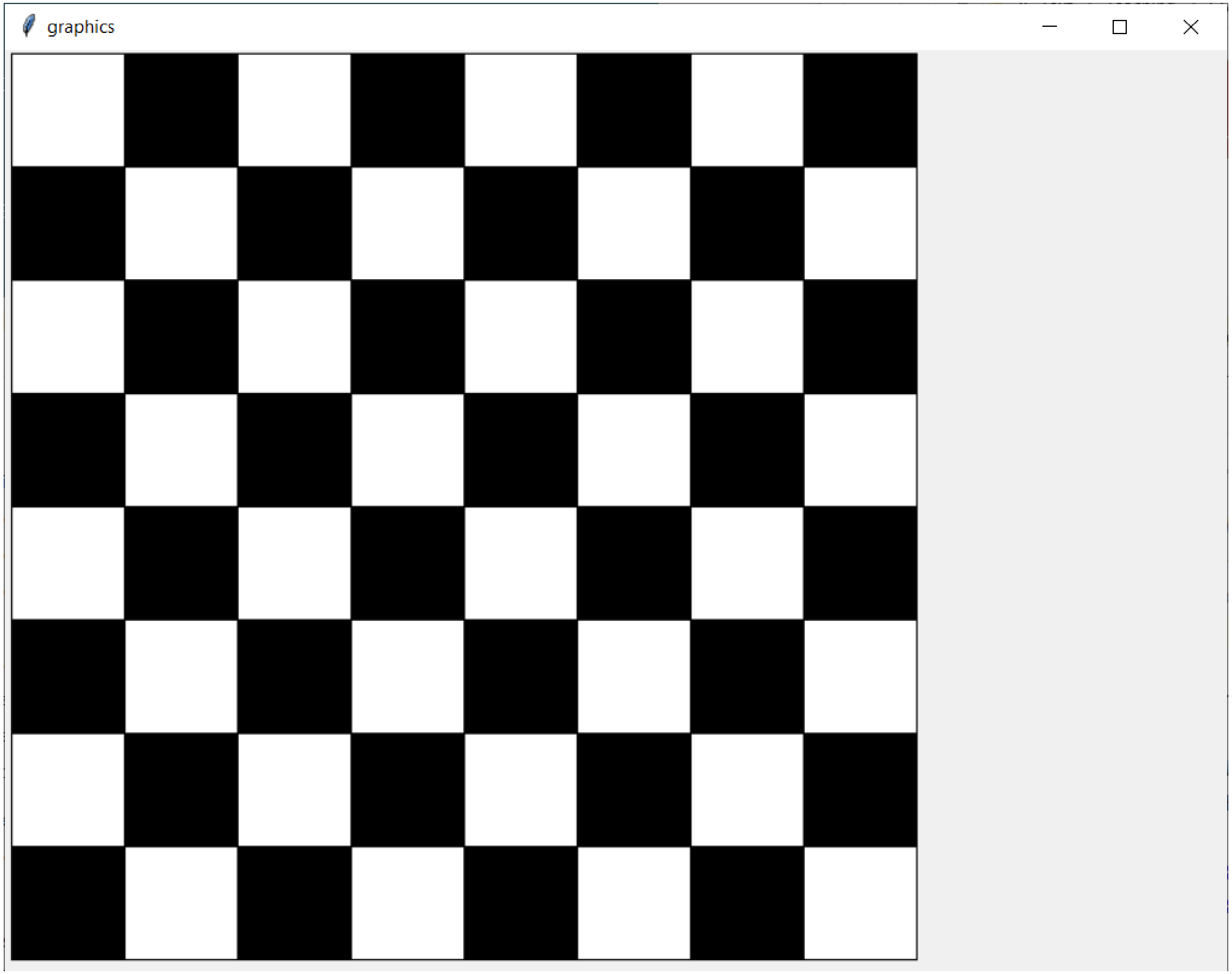
```
def draw_centered_rect(canvas, width, height, rect_fill=None):
    x = (CANVAS_WIDTH - width) / 2
    y = (CANVAS_HEIGHT - height) / 2
    canvas.create_rectangle(x, y, x + width, y + height,
        fill=rect_fill)

def drawing(canvas):
    draw_centered_rect(canvas, 200, 100)
    draw_centered_rect(canvas, 50, 25, rect_fill='blue')
```



Today's Goals

1. Learning about drawing basic graphics in Python
2. Creating programs that draw pictures

Putting It All Together



checkers.py

🎉🙌🙌