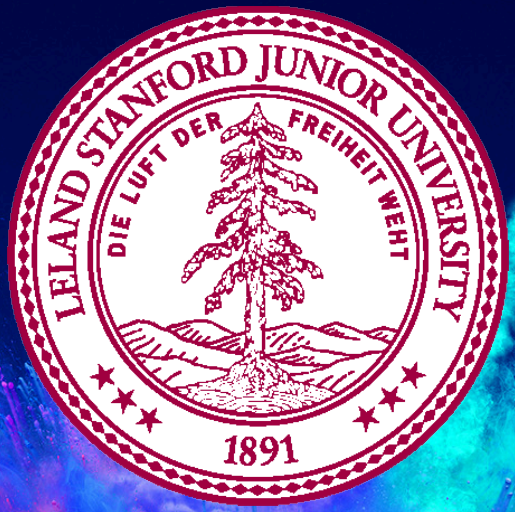
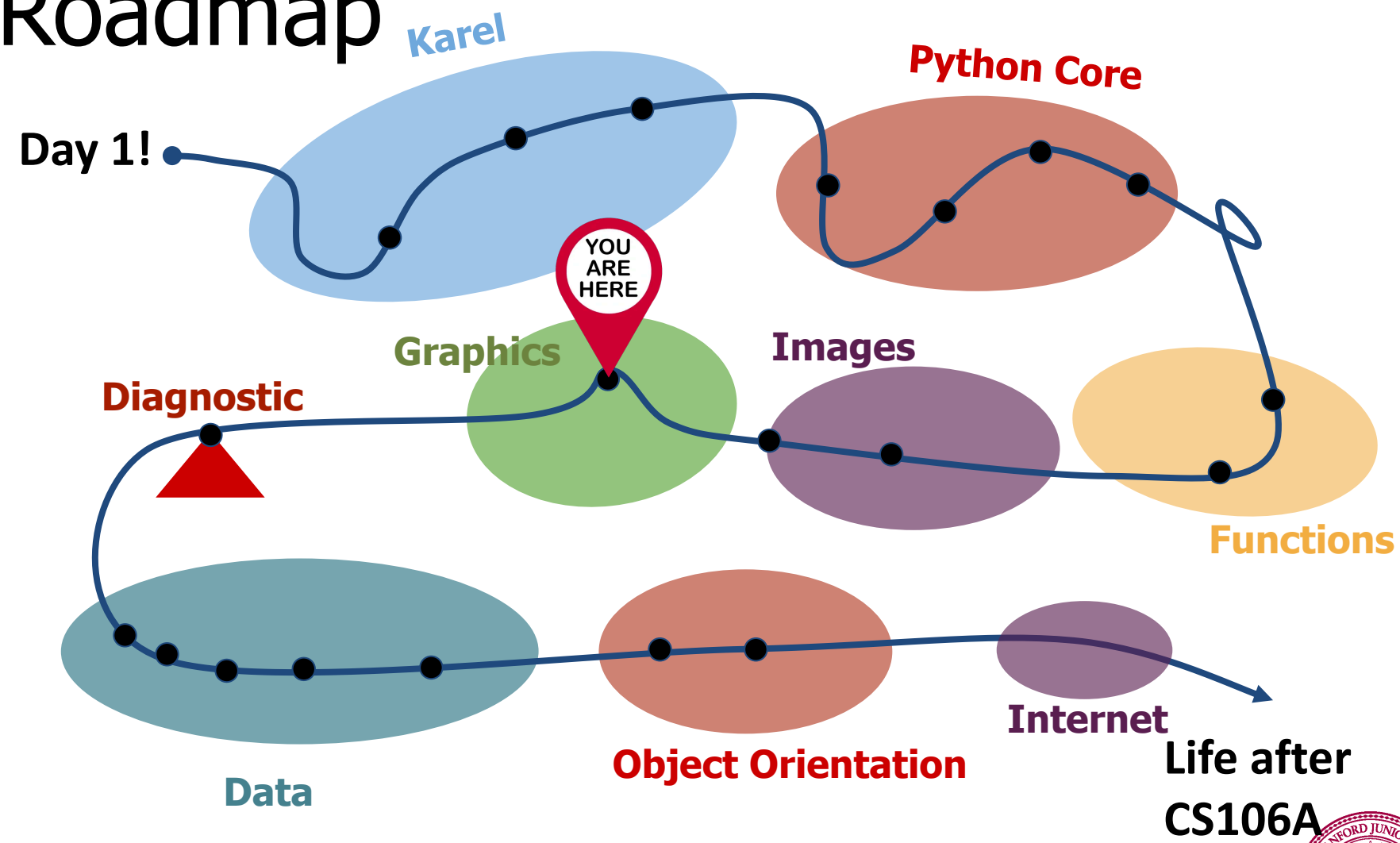


Graphics

Chris Gregg
Based on slides by Chris Piech and Mehran Sahami
CS106A, Stanford University

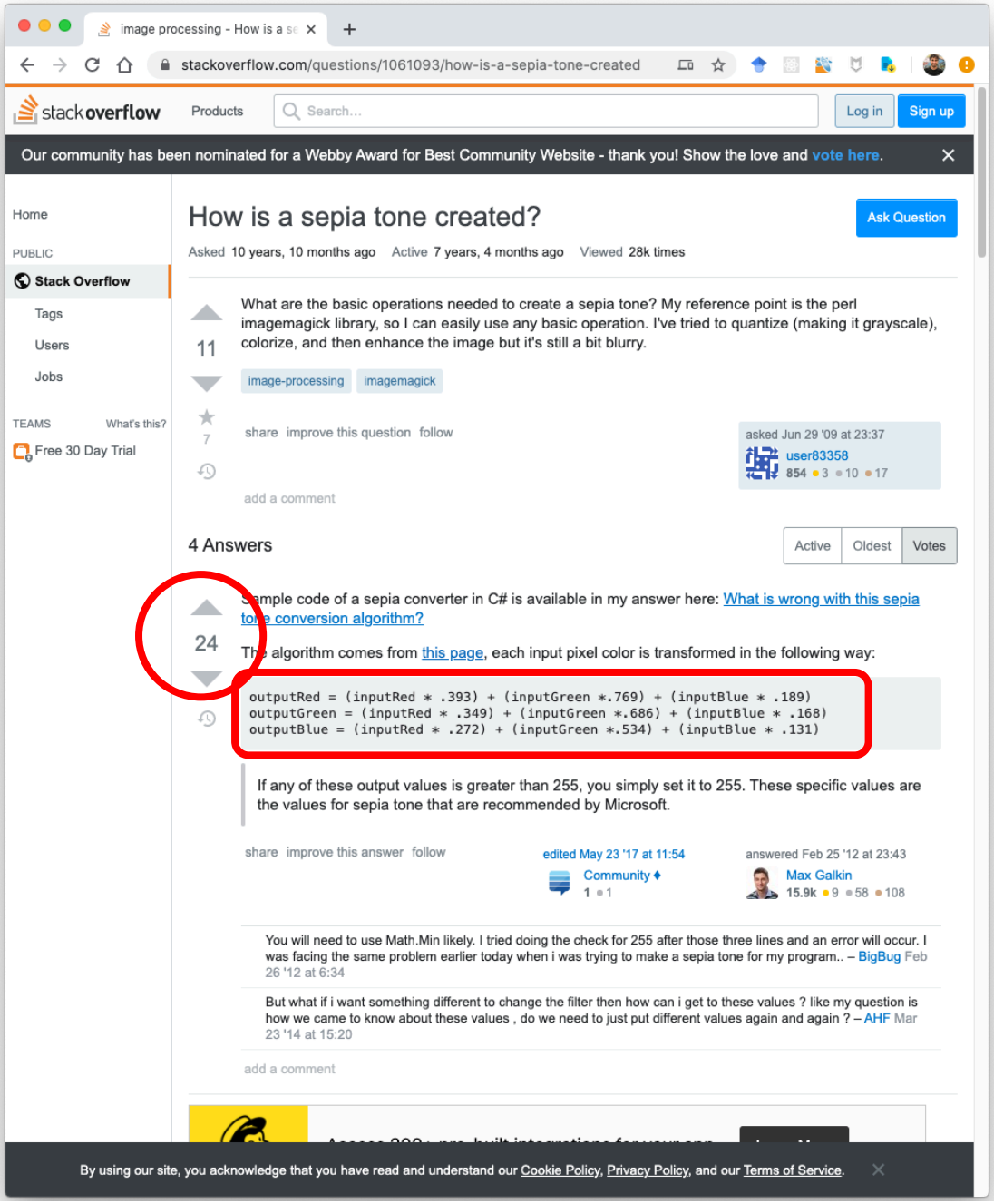
Roadmap


Karel
Python Core
Day 1!
YOU ARE HERE
Graphics
Images
Diagnostic
Functions
Internet
Object Orientation
Life after CS106A
Data

Assignment 3: Images and Graphics!

- You can find assignment 3 on the CS 106A website
- It is a challenging assignment, but also a lot of fun
- There are two image problems (a warmup and a longer problem) and one graphics problem made up of a number of smaller problems.
- We will take some time now to go over the different problems.
- Don't wait to start this assignment! Even though we have the diagnostic on Thursday, you should start the assignment now. Even though images and graphics won't be on the diagnostic, the programming you will do for this assignment will give you more Python practice and will help you study.

Review



https://stackoverflow.com/questions/1061093/how-is-a-sepia-tone-created

Sepia Example

```
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for pixel in image:
        sepia_pixel(pixel)
    image.show()
```

```
def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
```

Sepia Example

```
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
```

Sepia Example

```
def main():
    image_name = input('enter an image name: ')
    image = SimpleImage('images/' + image_name)
    for y in range(image.height):
        for x in range(image.width):
            pixel = image.get_pixel(x, y)
            sepia_pixel(pixel)
    image.show()

def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
```

Sepia Example

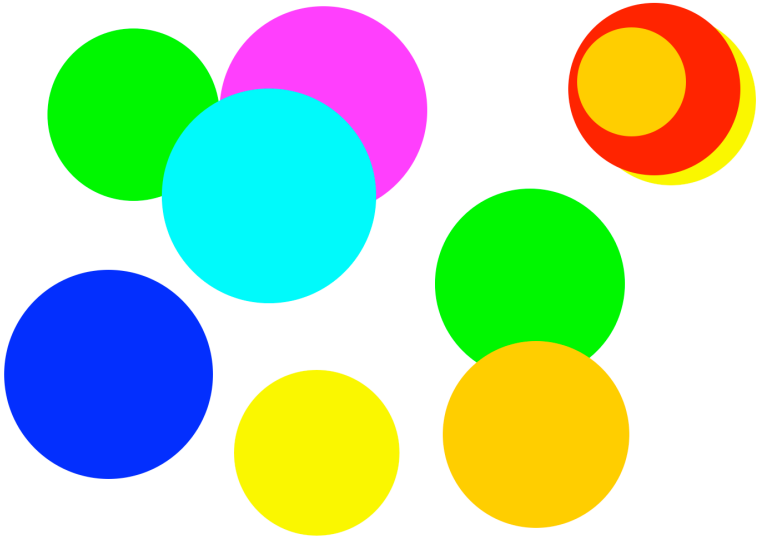
```
def main():


    for y in range(600):
        for x in range(800):
            print(x, y)
```

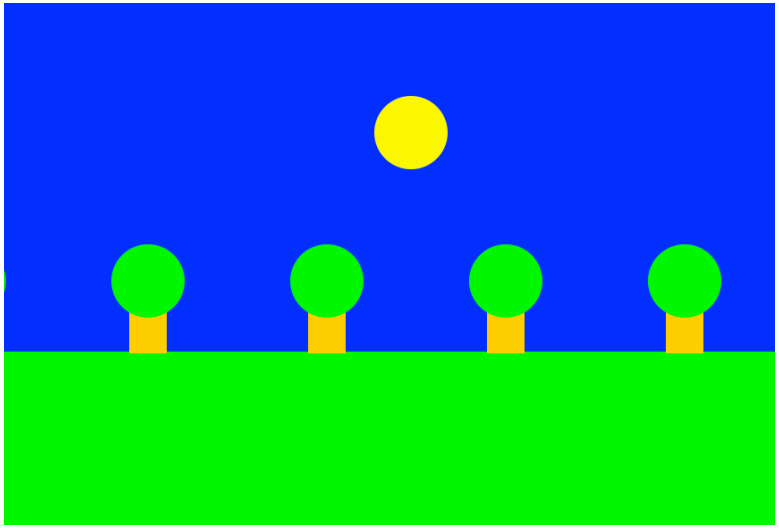
End Review

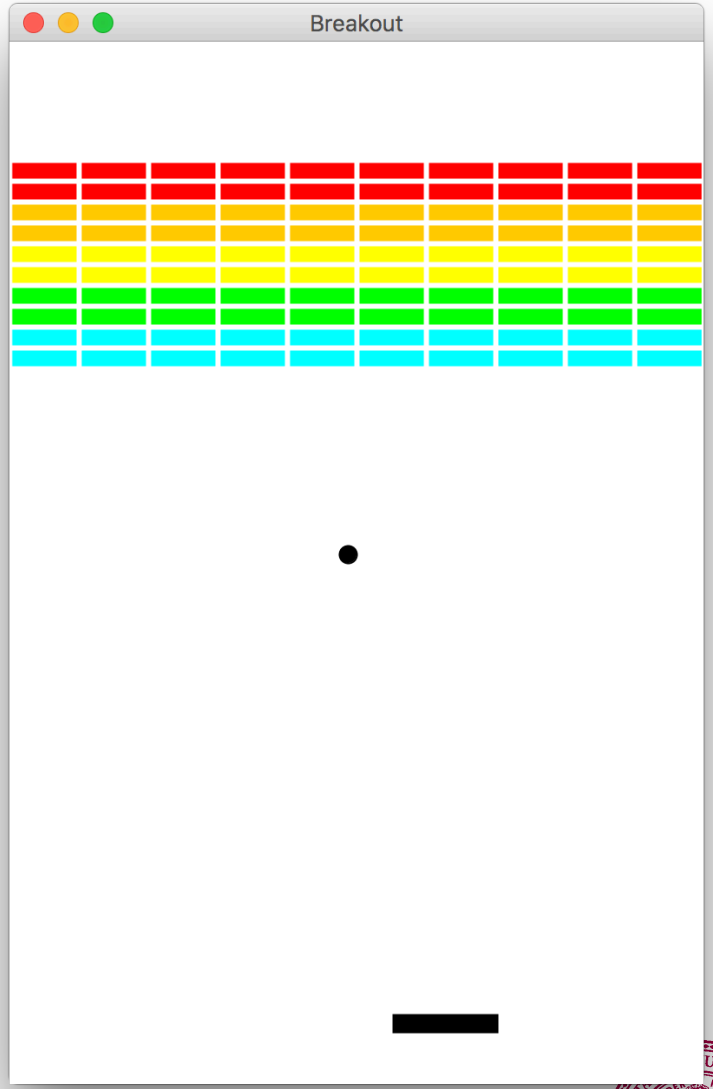
Today's Goal

1. How do I draw shapes?

Graphics Programs



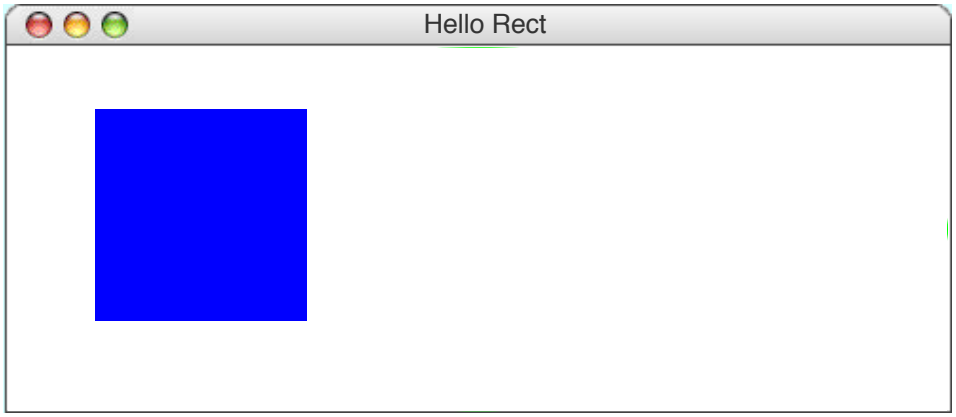




Draw a Rectangle

the following `main` method displays a blue square

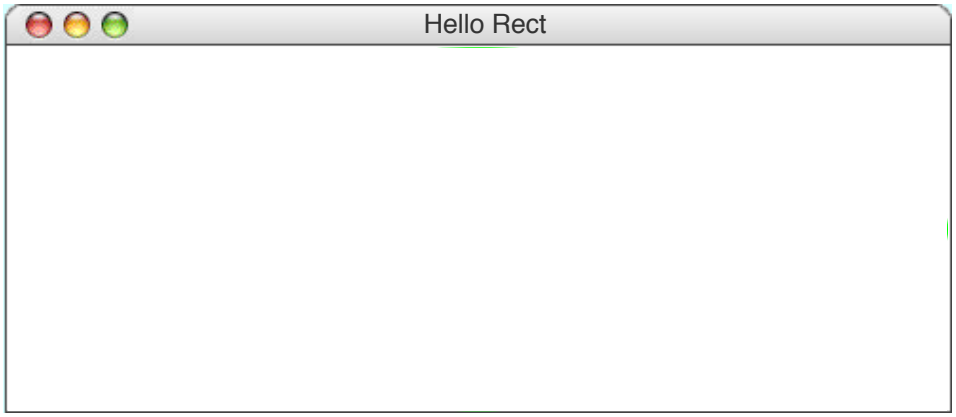
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```



Draw a Rectangle

the following **`main`** method displays a blue square

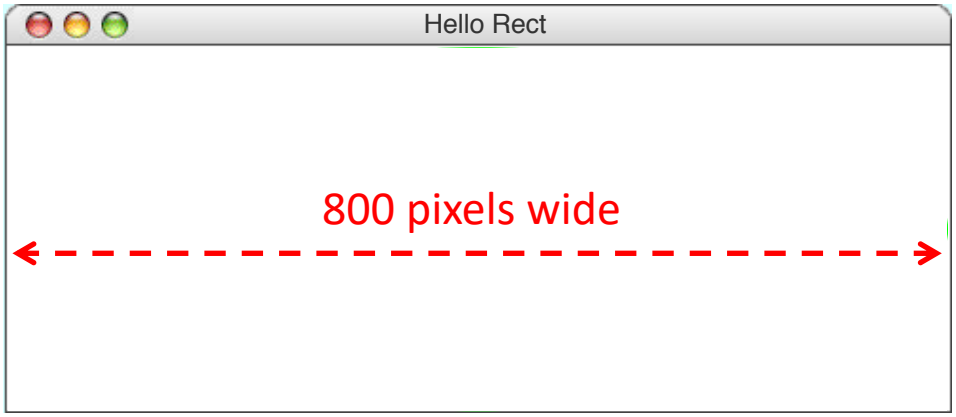
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

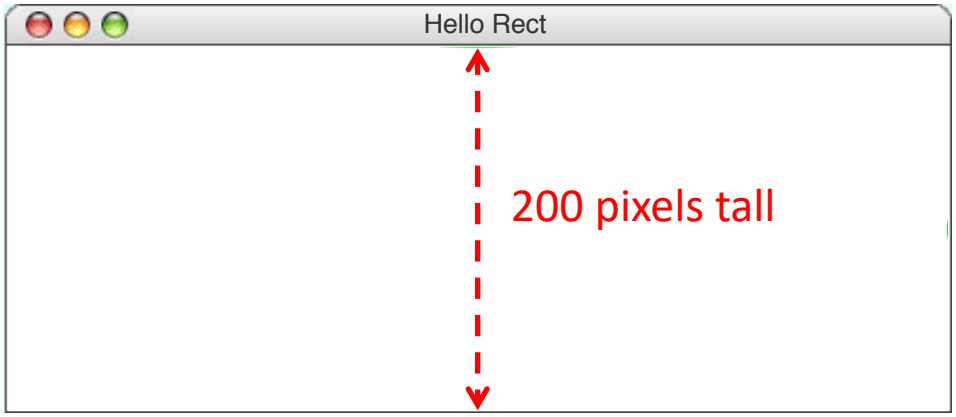
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following **`main`** method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

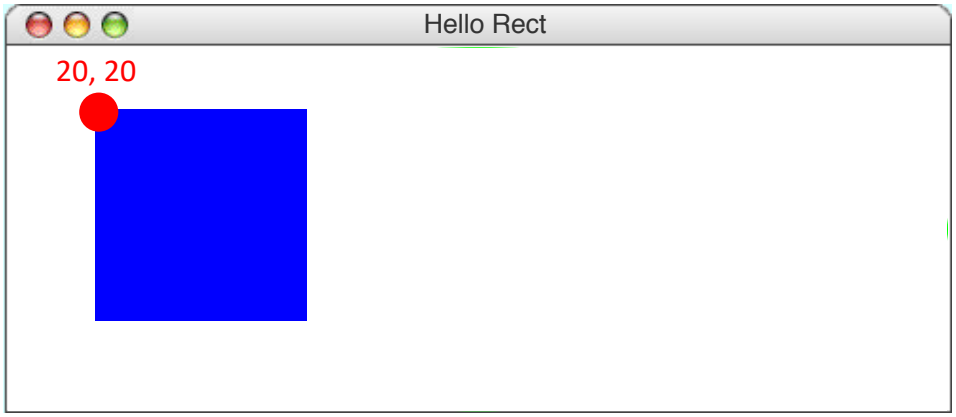
the following **`main`** method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```

Draw a Rectangle

the following `main` method displays a blue square

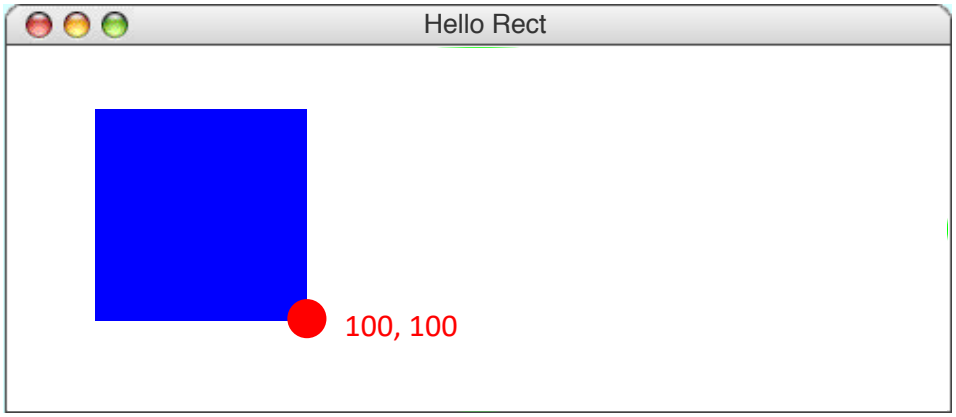
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

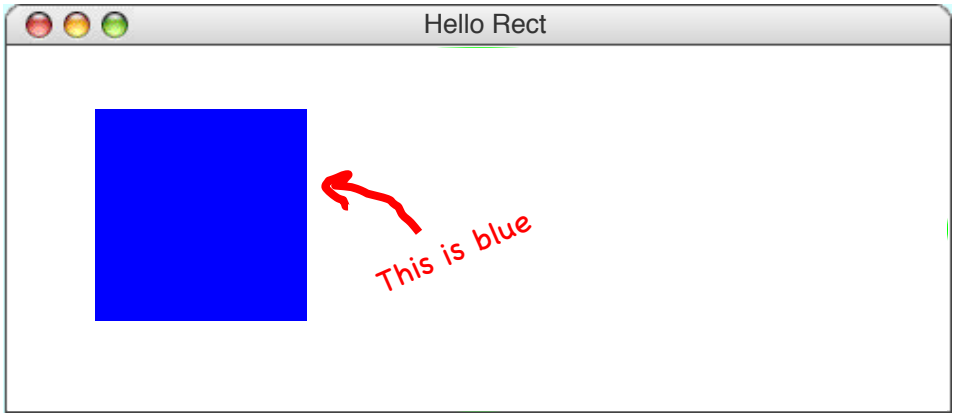
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



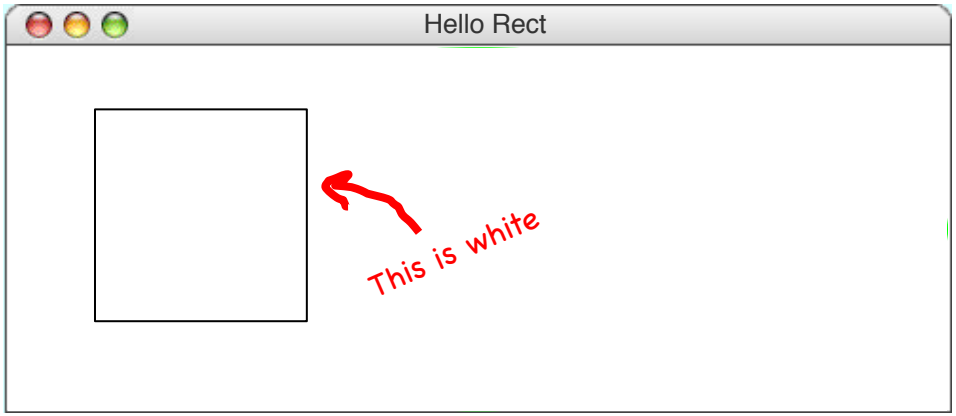
Aside: Named Arguments

This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

the following `main` method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100)
```

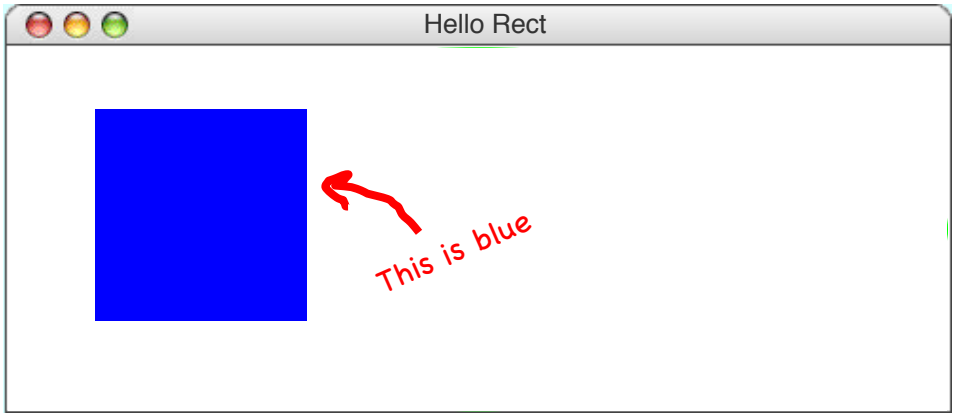


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

the following `main` method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



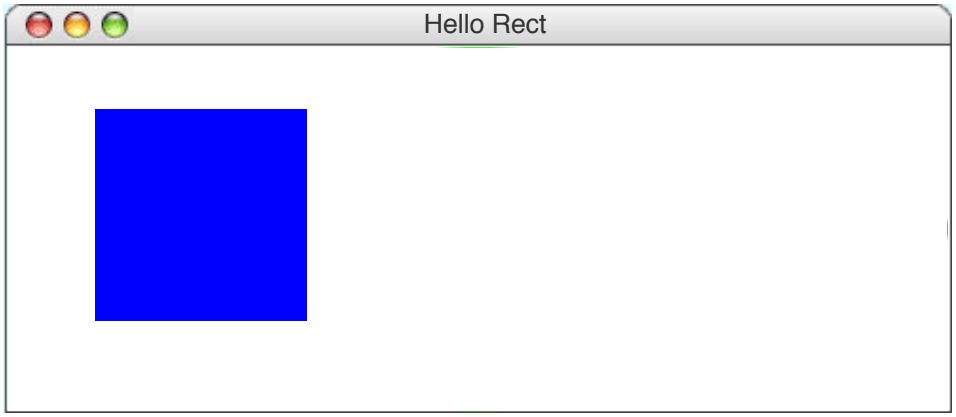
Aside: Named Arguments

This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

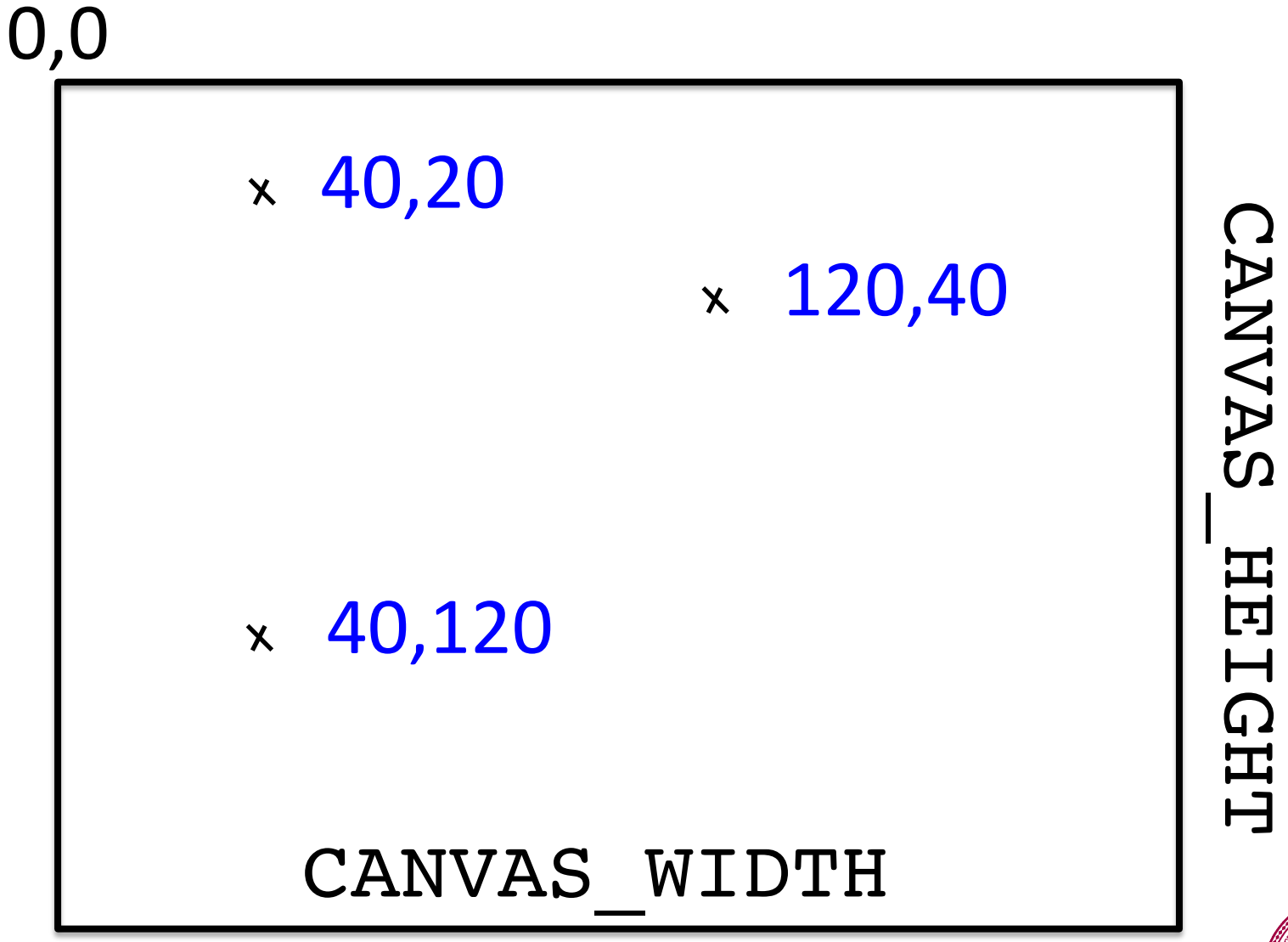
the following `main` method displays a blue square

```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```

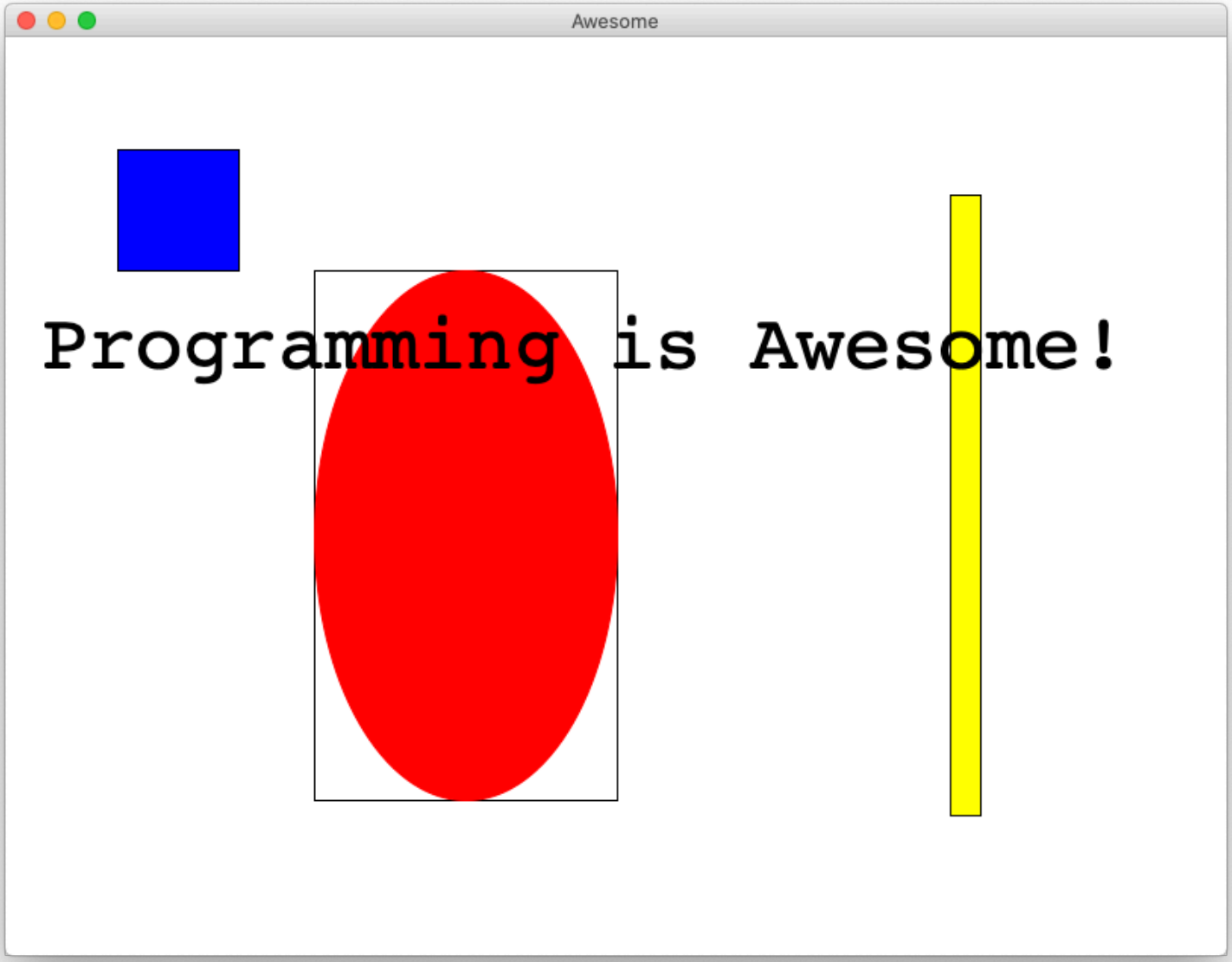


TK Natural Graphics

Graphics Coordinates



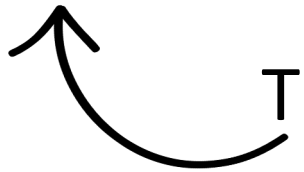
Rectangles, Ovals, Text



- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text()`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`



The first point of the line is `(x1, y1)`

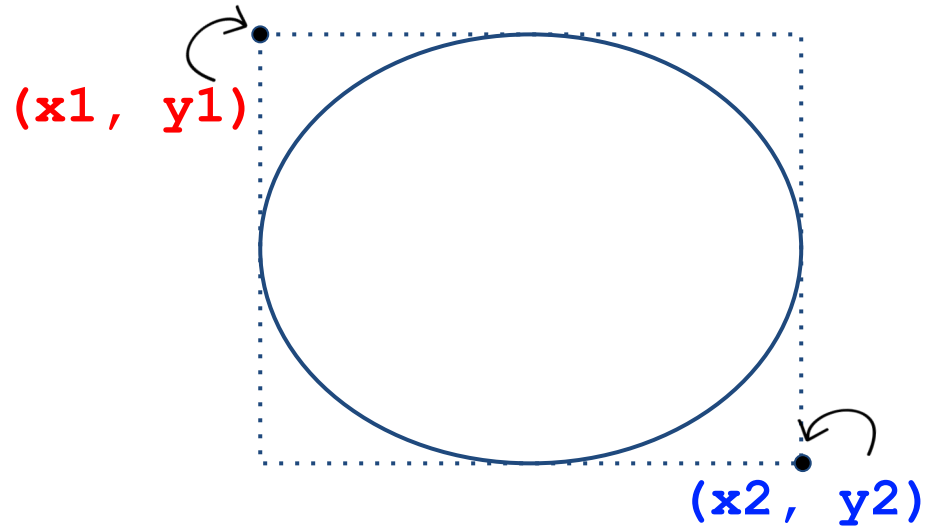
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

The second point of the line is `(x2, y2)`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

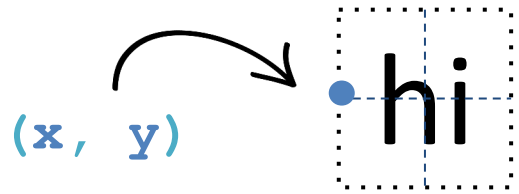
- `canvas.create_line()`
- **`canvas.create_oval(x1, y1, x2, y2)`**
- `canvas.create_text()`

- canvas.create_line()
- **canvas.create_oval(x1, y1, x2, y2)**
- canvas.create_text()



- `canvas.create_line()`
- `canvas.create_oval()`
- **`canvas.create_text(x, y, text='hi')`**

- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text(x, y, text='hi', anchor='w')`



Pedagogy

CS106A

localhost:8000/examples/awesome/

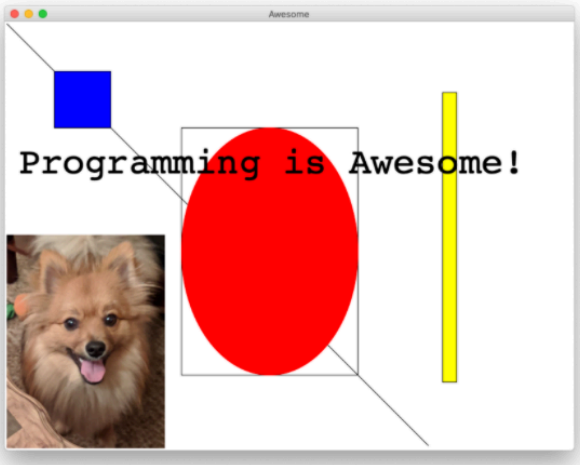
CS106A Lectures Assignments Section Handouts Examples Schedule

Programming is Awesome

Programming is Awesome

BY CHRIS PIECH

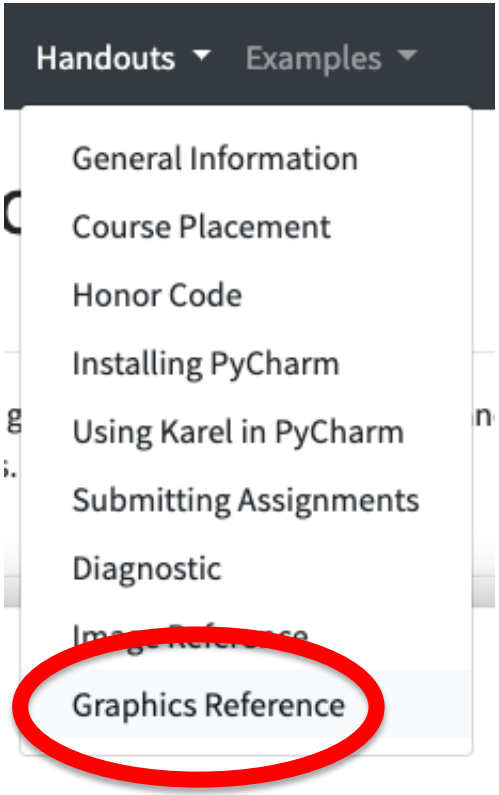
Graphics are really fantastic in python, especially using the TK library (which is the standard for Python). There are a lot of details, and as such a great way to learn is to look at worked examples.



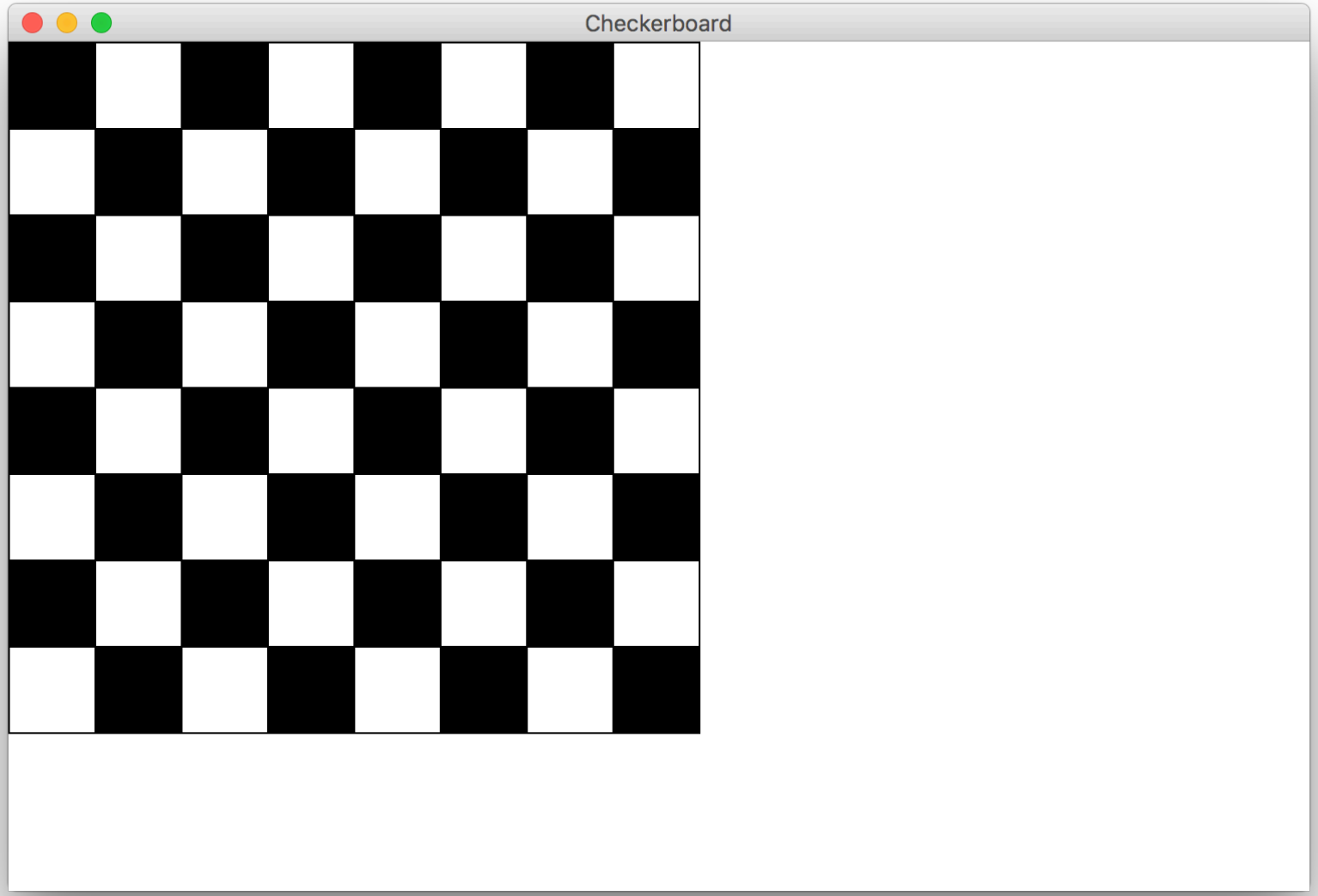
Solution

```
import tkinter
from PIL import ImageTk
from PIL import Image

CANVAS_WIDTH = 800
CANVAS_HEIGHT = 600

def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Awesome')
    # a line for good measure!
    canvas.create_line(0, 0, 600, 600)

    # a blue square with width and height = 80
    canvas.create_rectangle(70, 70, 150, 150, fill="blue")
    # a yellow rectangle that is long and skinny
    canvas.create_rectangle(620, 100, 640, 510, fill="yellow")
```

localhost:8000/examples/awesome/

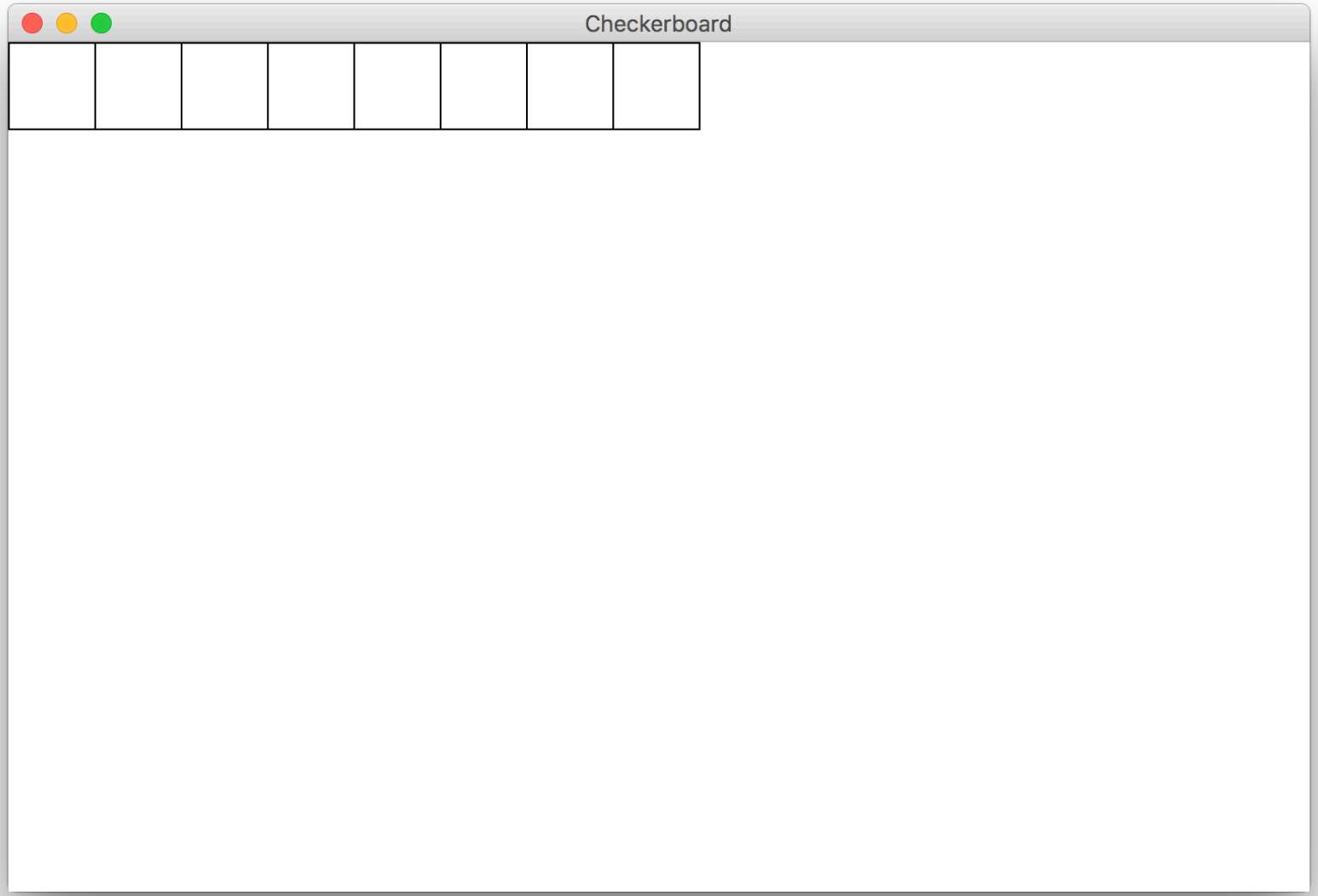


Goal

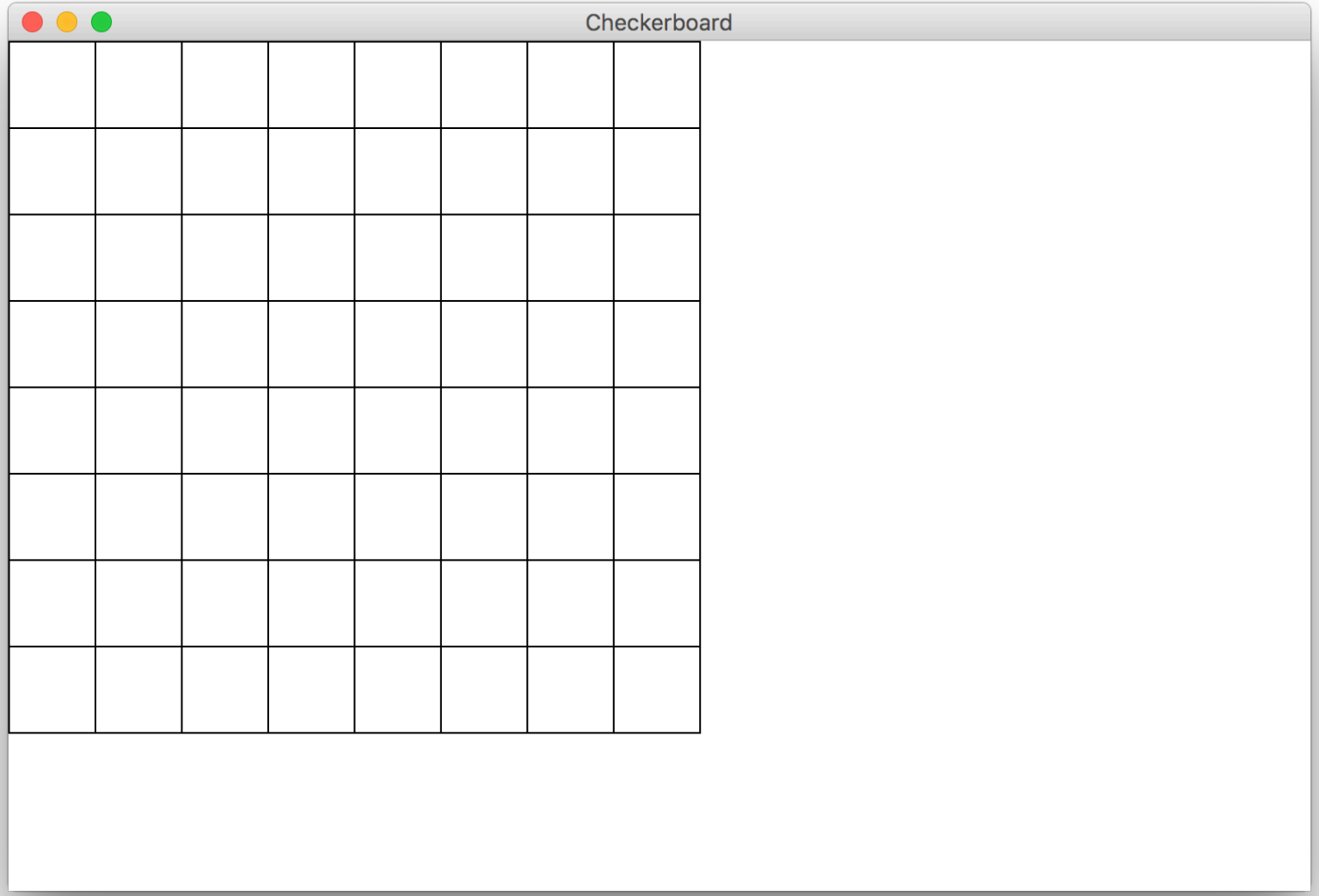

Checkerboard


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Milestone 1

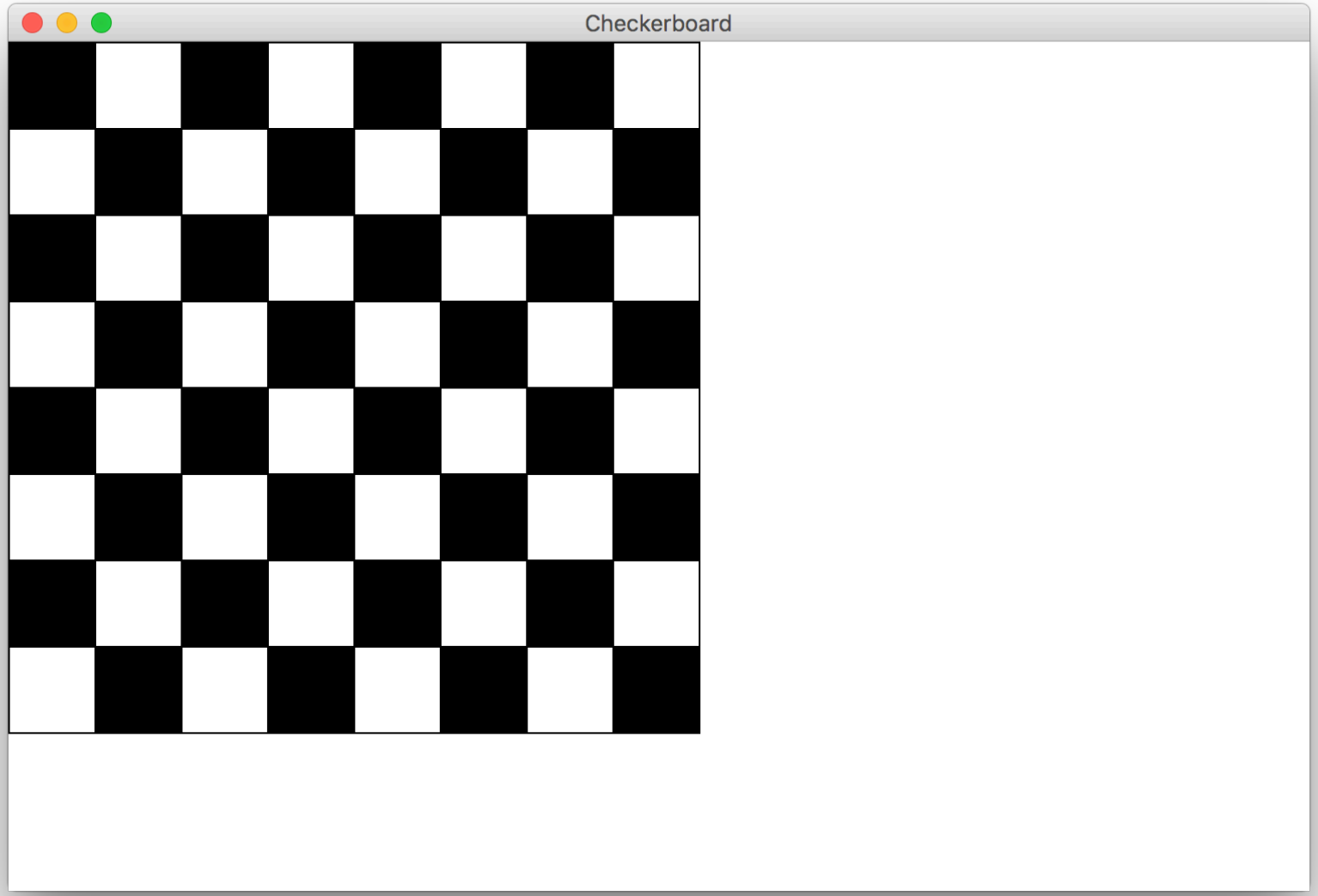


Milestone 2

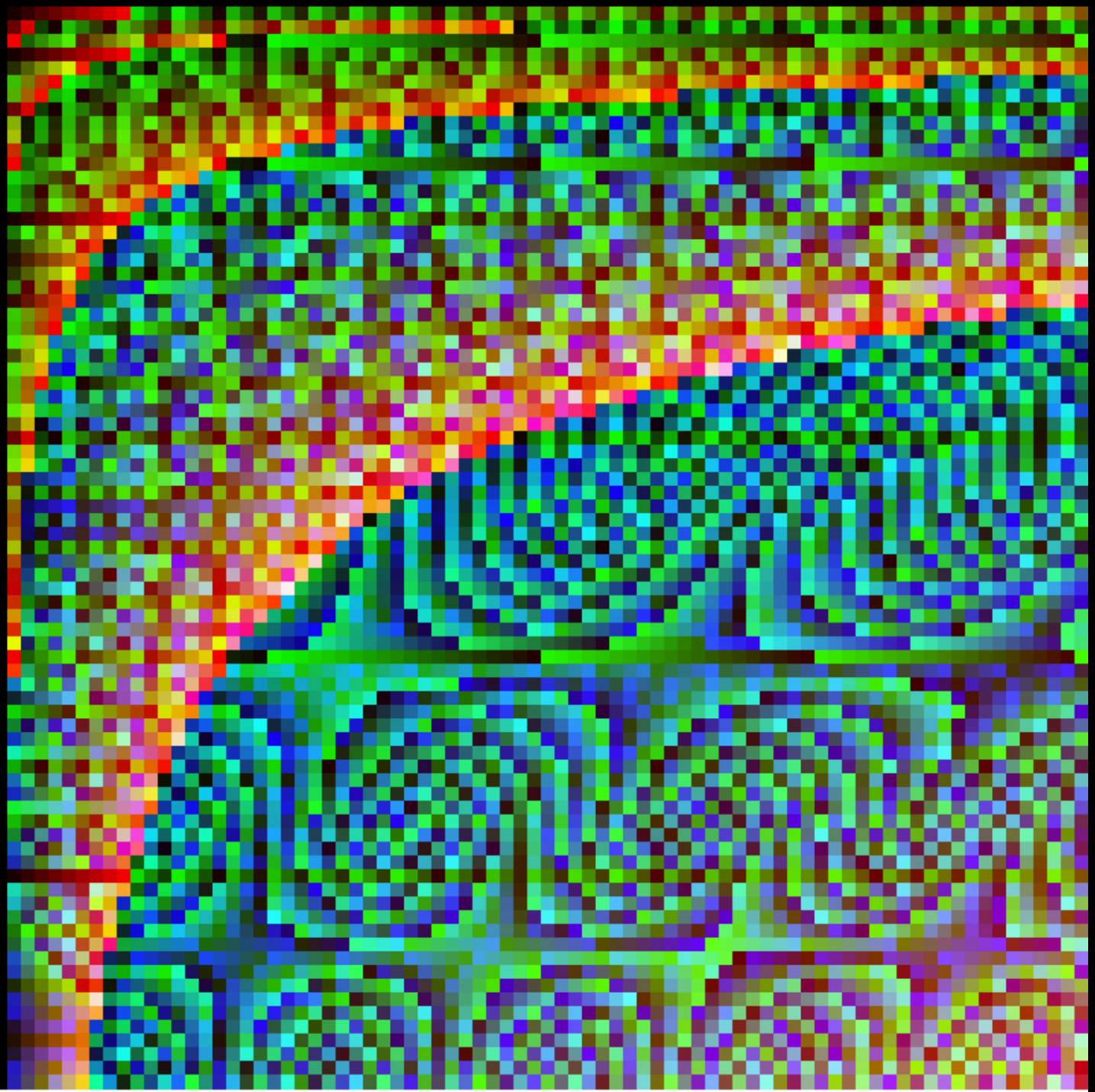

Checkerboard


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891

Milestone 3


Checkerboard


LELAND STANFORD JUNIOR UNIVERSITY
DIE LUFT DER FREIHEIT WEHT
1891



Teaser for tomorrow…

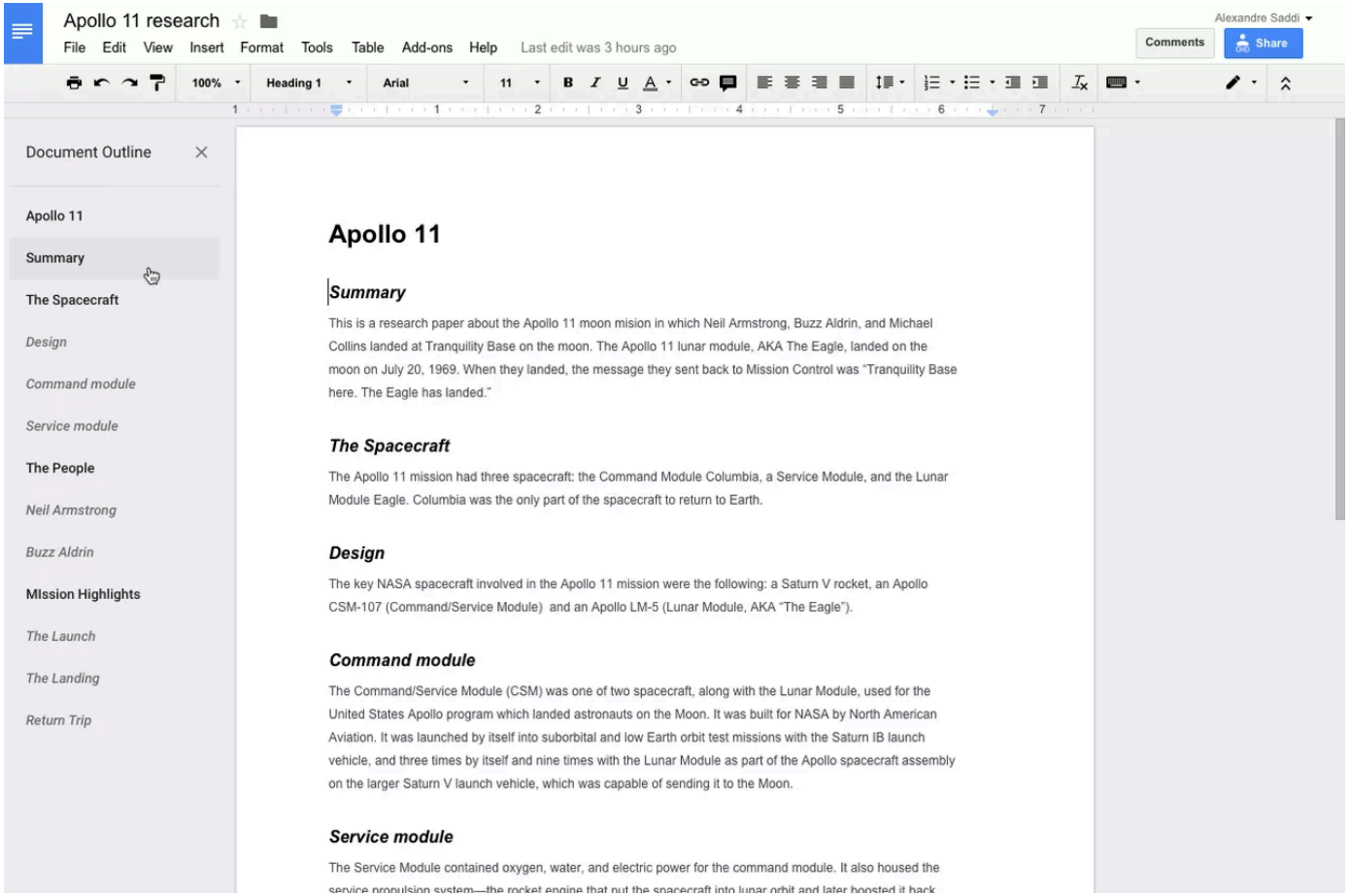
Hold up!

```
def draw_square(canvas, row, col):
```

If you get a copy when you pass a parameter. Does this copy the canvas??!!

Large variables are stored using something like a URL. The URL gets copied

How do you share google docs?



https://docs.google.com/document/d/1eBtnEiII3KHefFS-kSAOpXqeSXpbfTTMlmOgj6I9dvk/

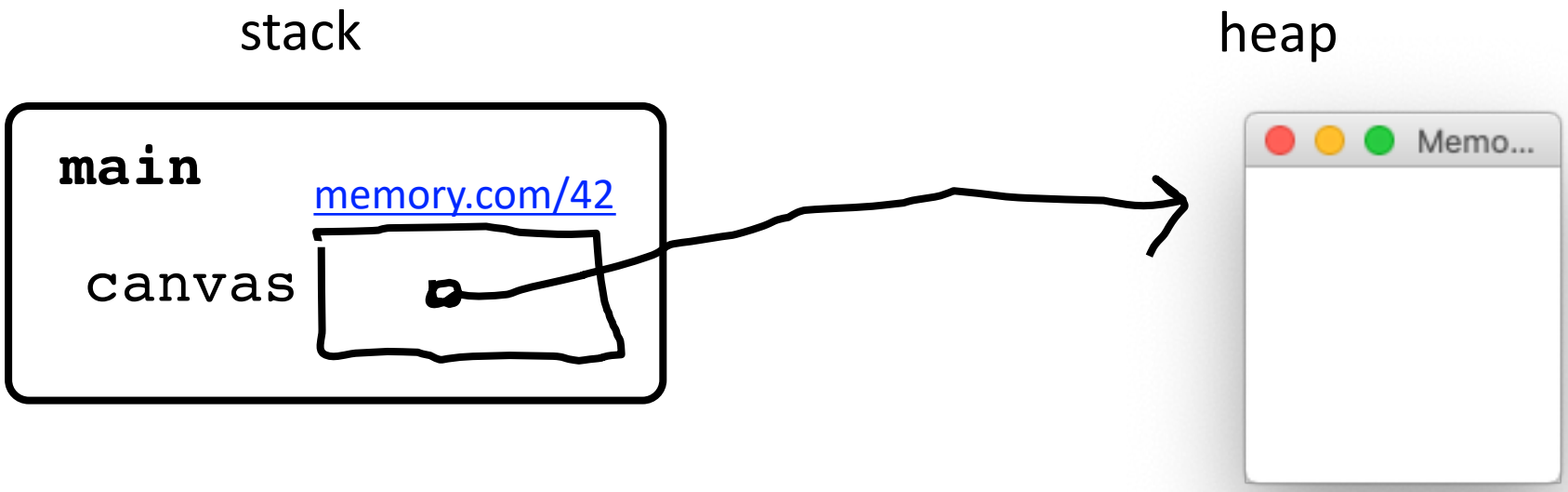
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```

---

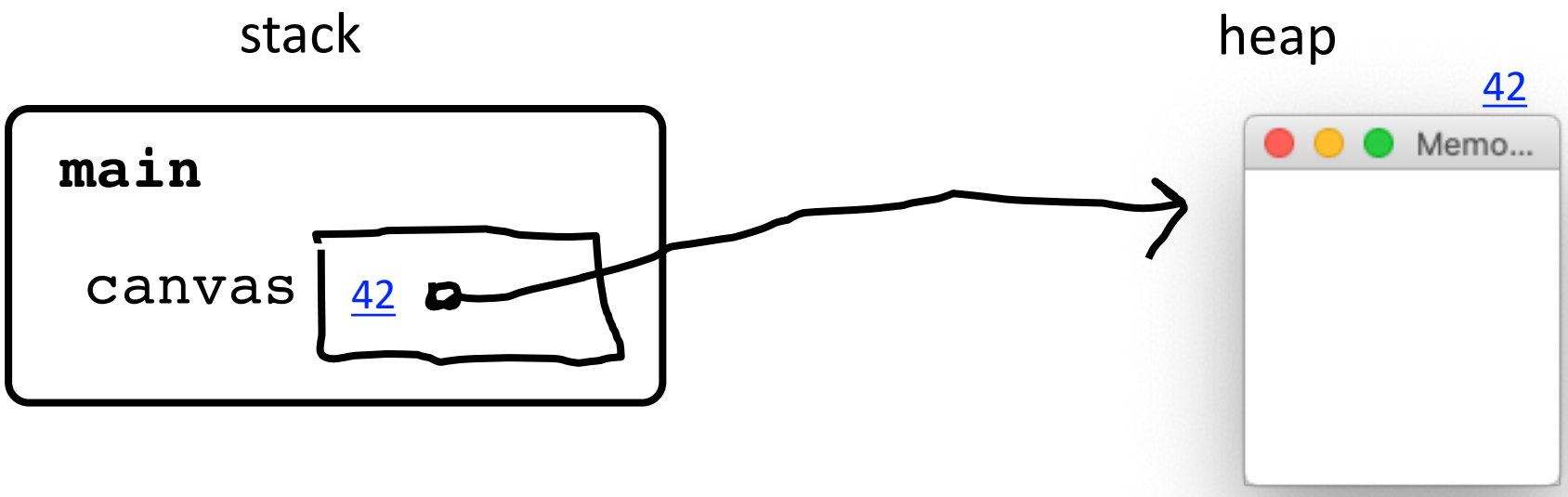
stack

heap

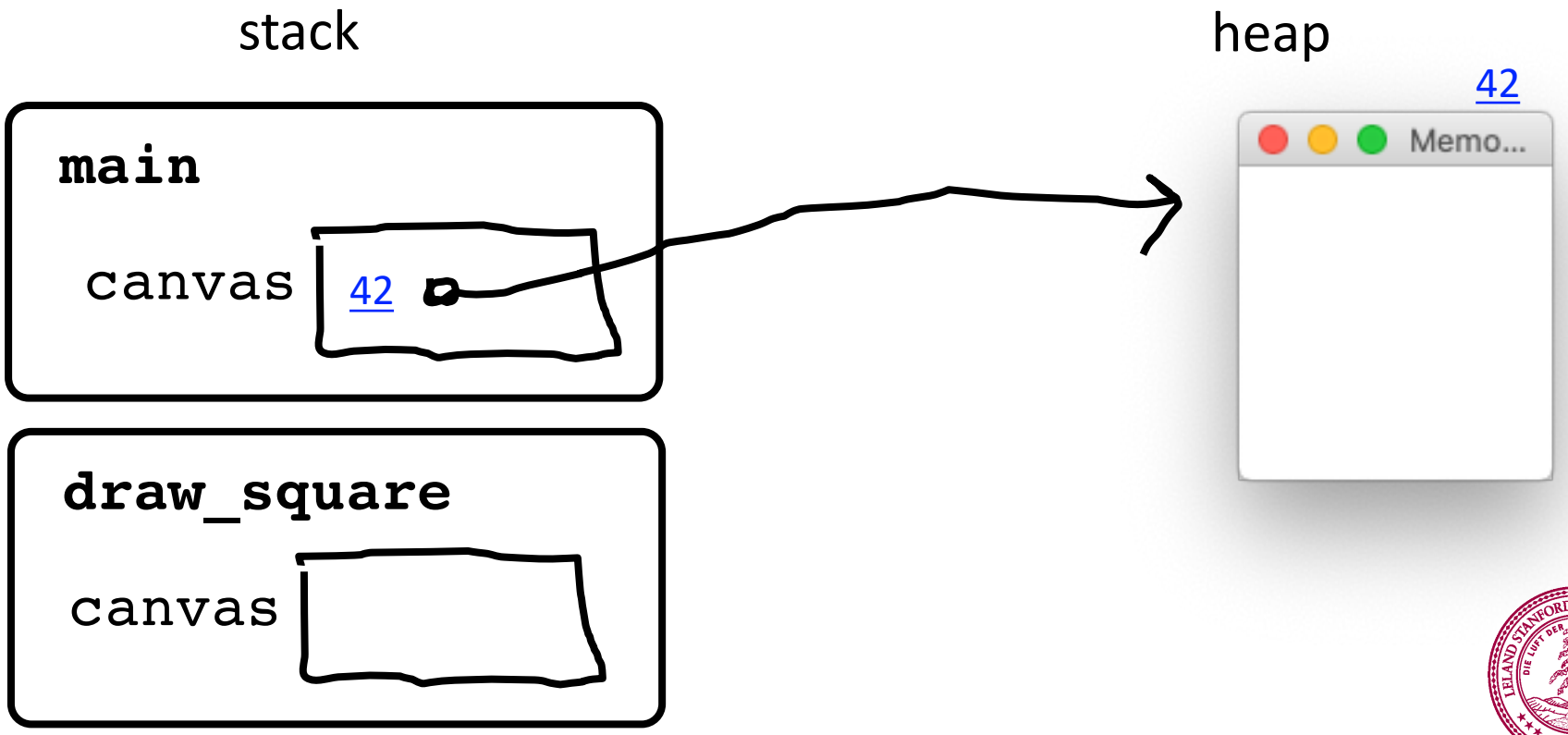
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



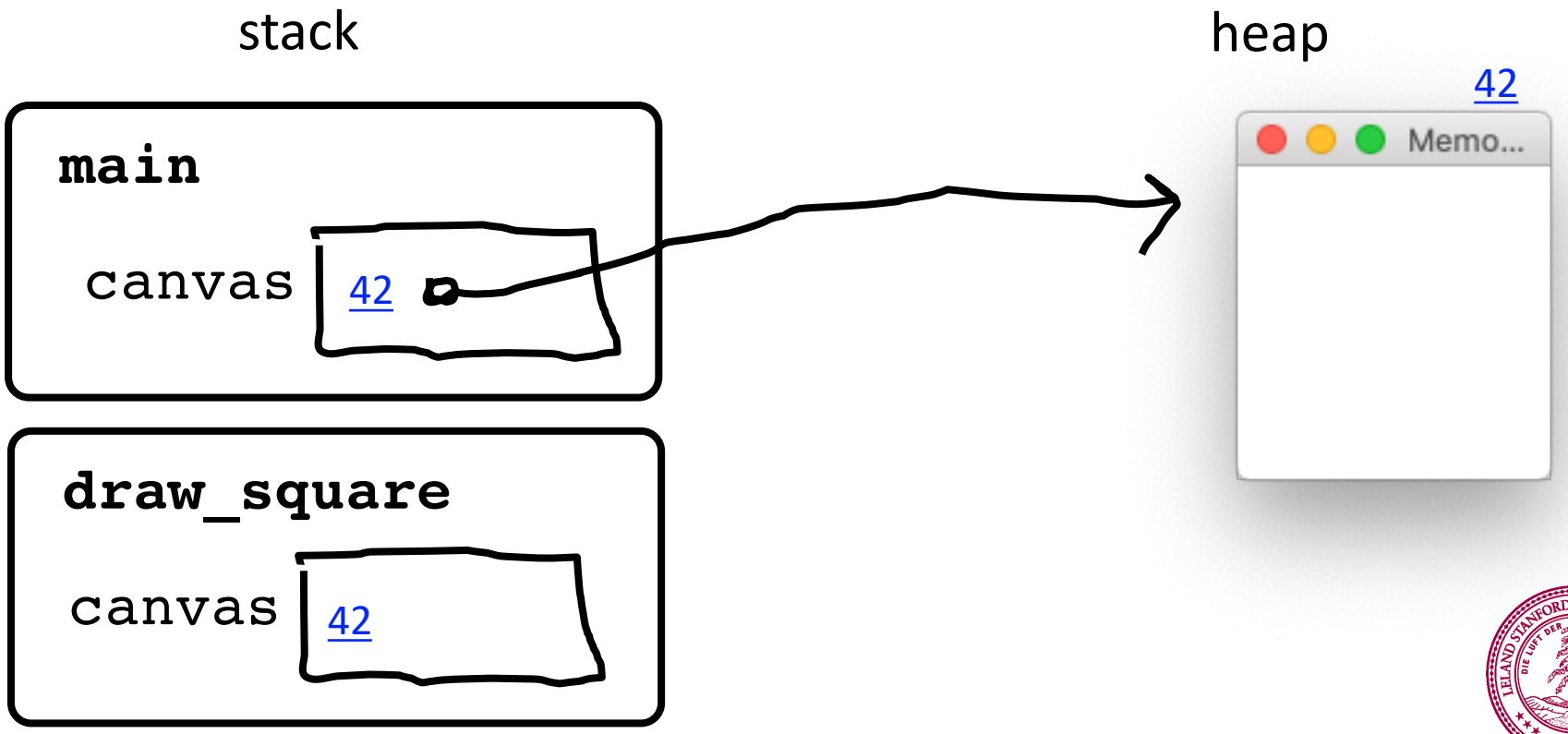
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



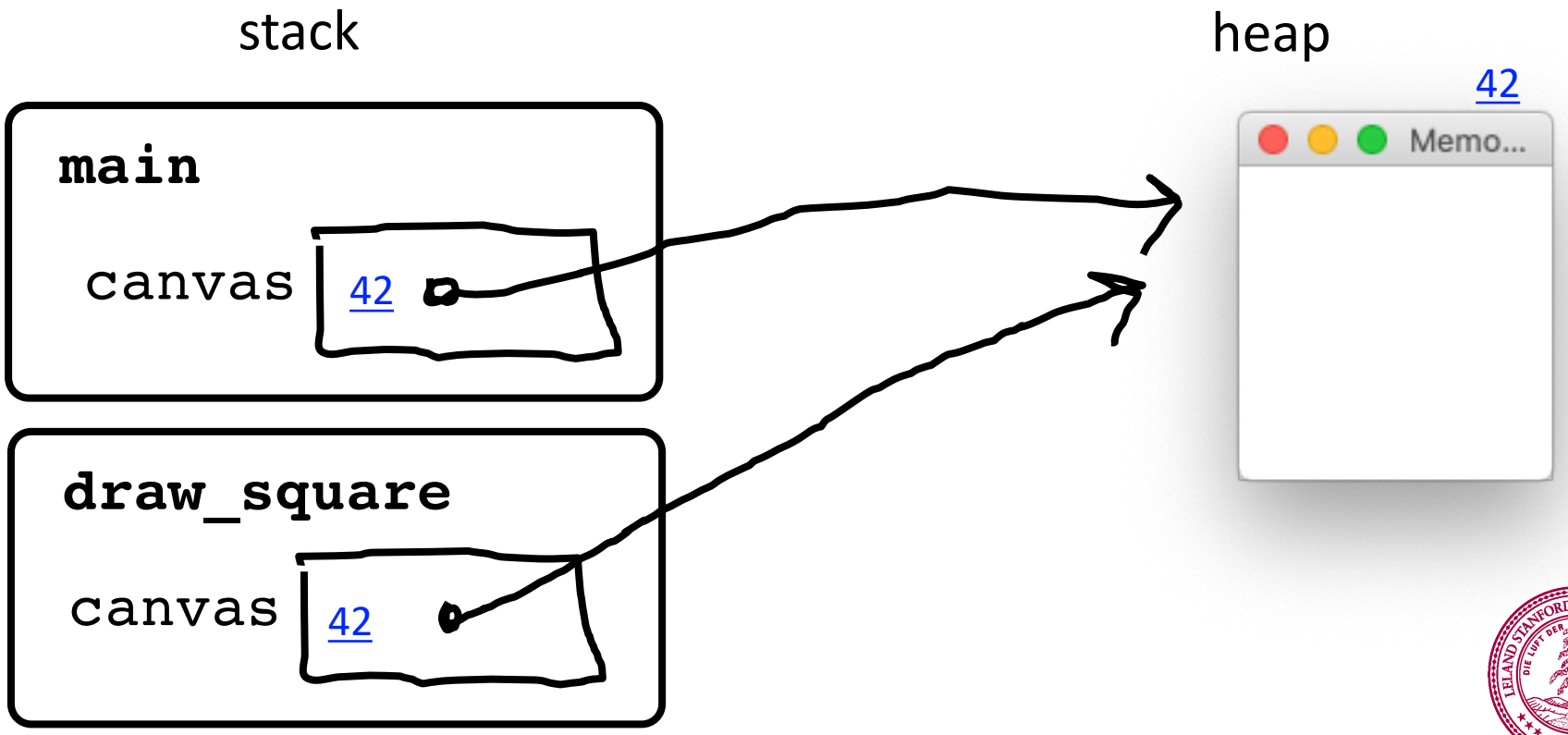
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



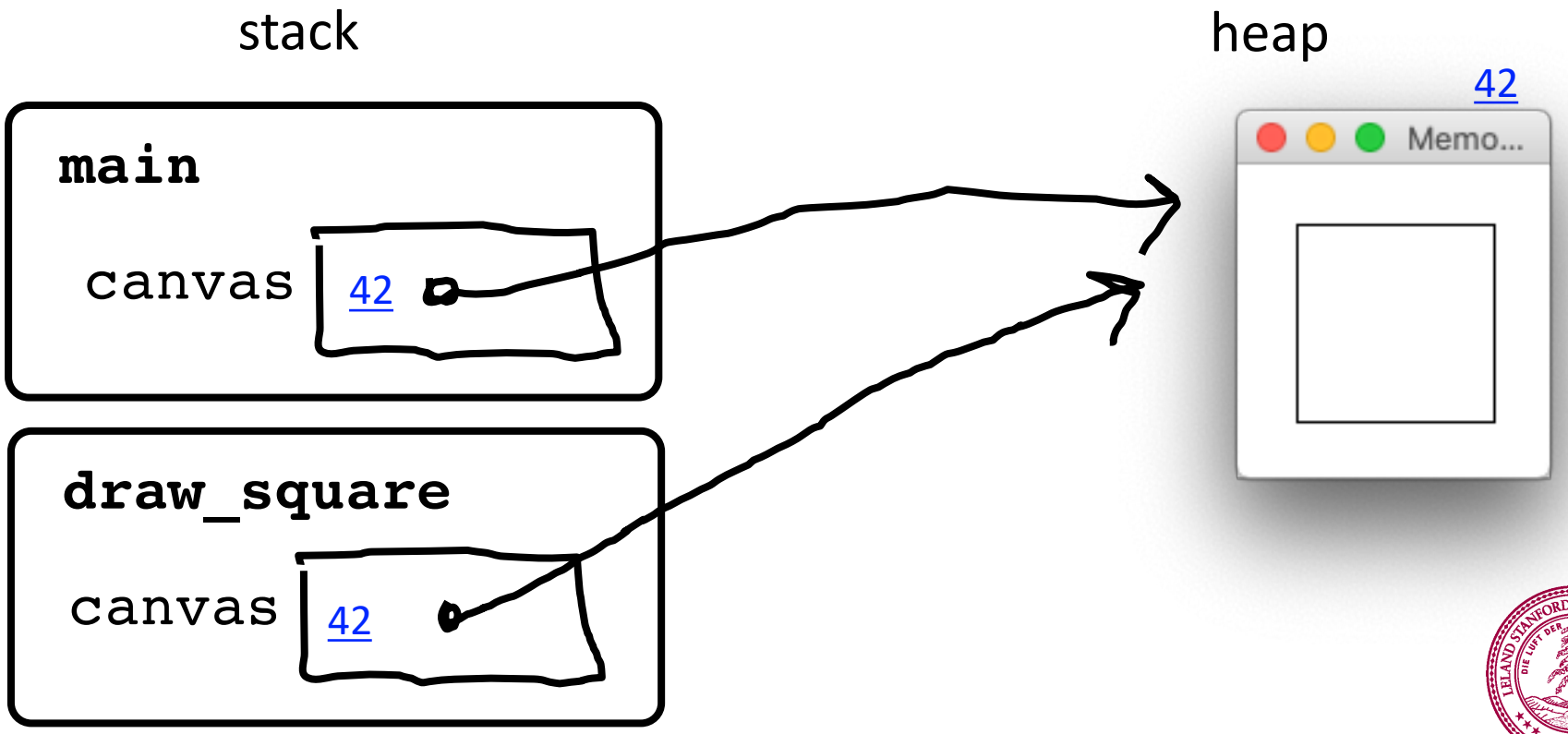
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



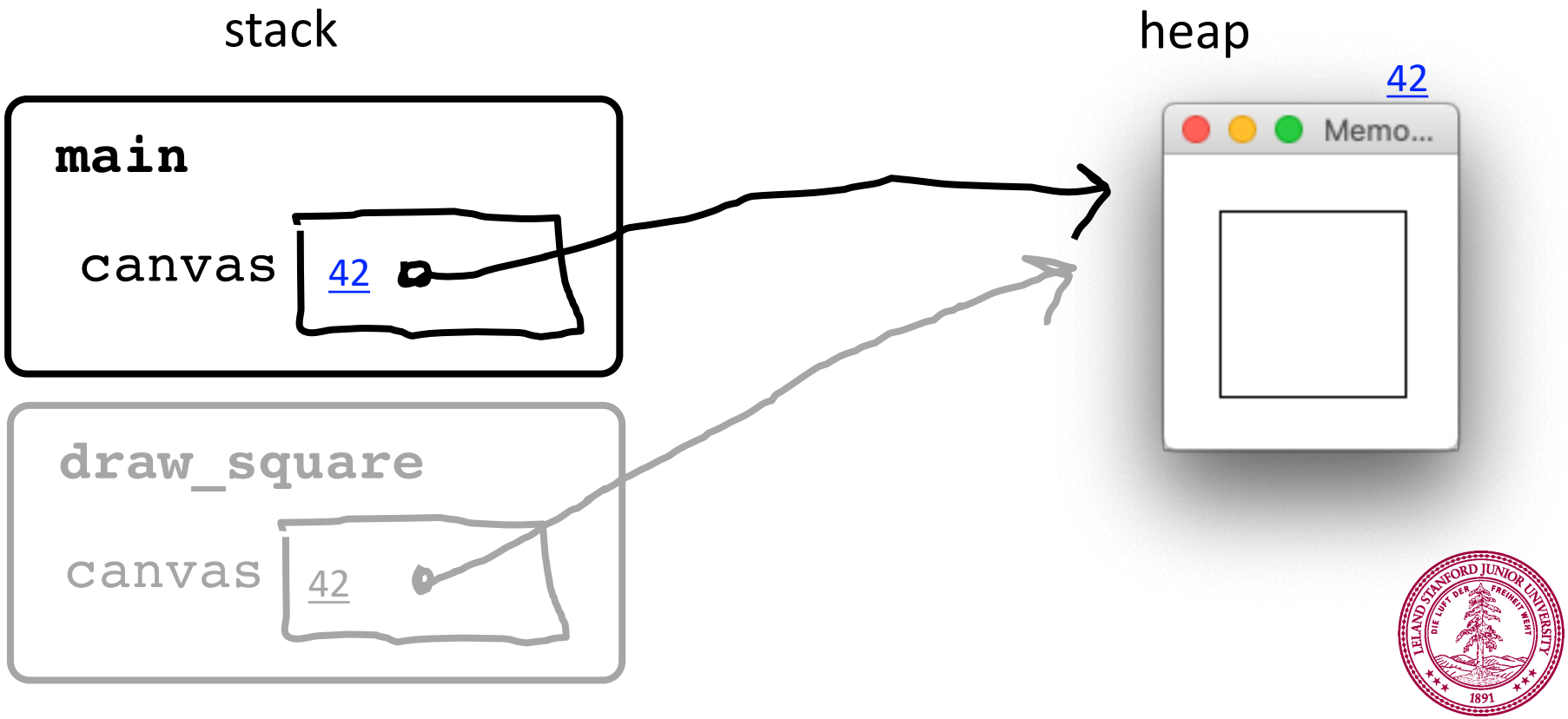
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



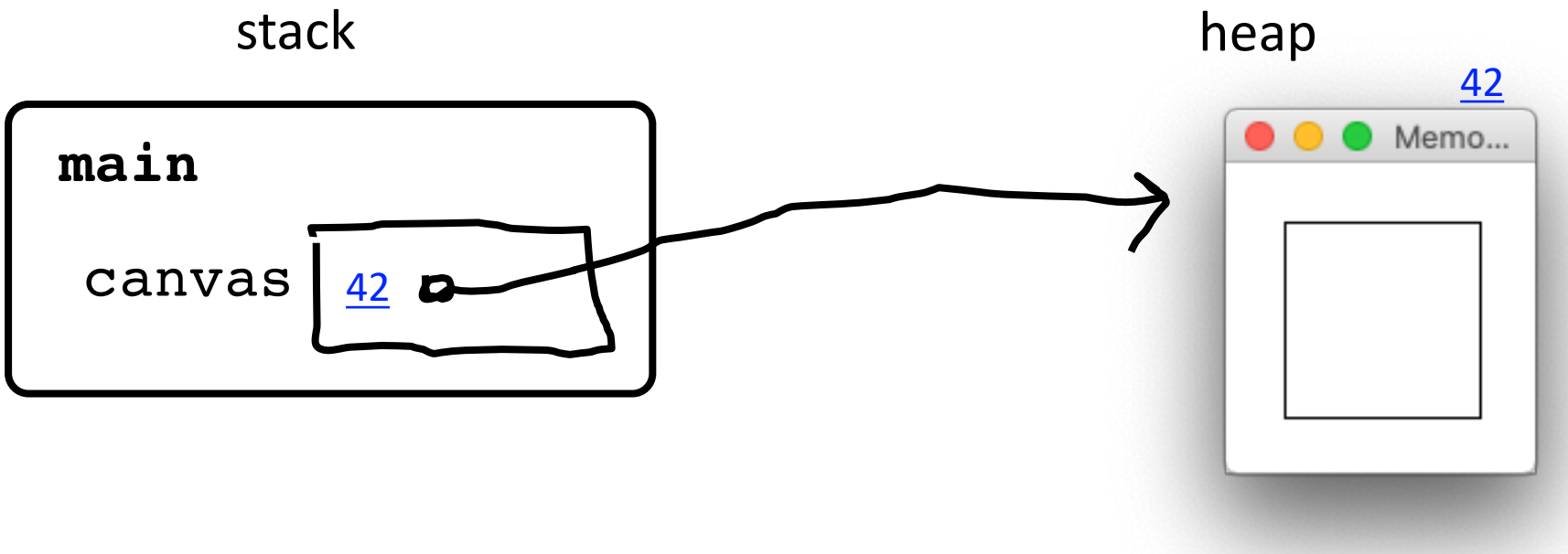
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



Large variable types are stored as memory addresses

(which are like memory URLs)