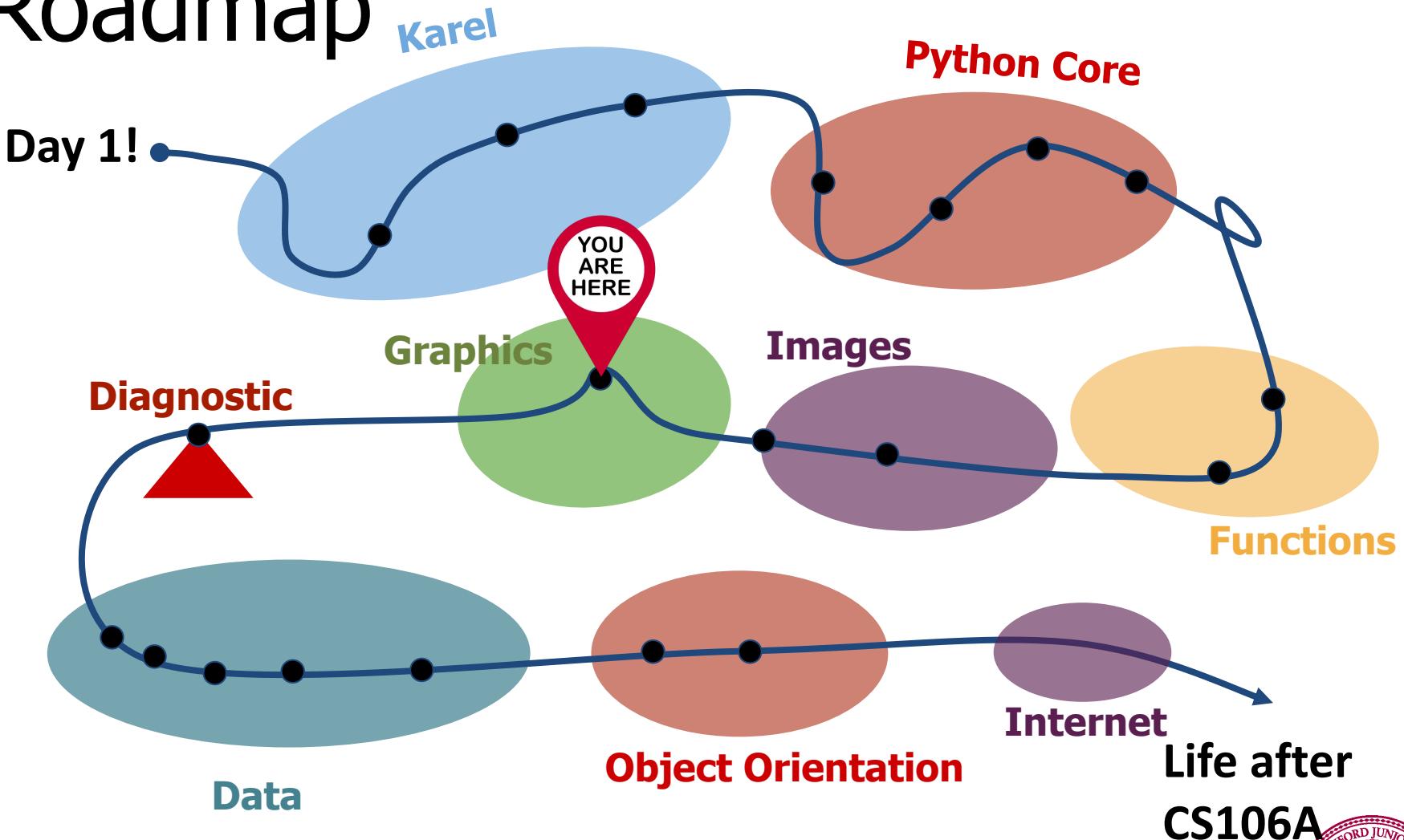


Graphics

Chris Gregg

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

Roadmap



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





image processing - How is a sepi... +

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

stackoverflow Products Search... Log in Sign up

Our community has been nominated for a Webby Award for Best Community Website - thank you! Show the love and [vote here](#). X

Home PUBLIC Stack Overflow Tags Users Jobs TEAMS What's this? Free 30 Day Trial

How is a sepia tone created?

Asked 10 years, 10 months ago Active 7 years, 4 months ago Viewed 28k times Ask Question

What are the basic operations needed to create a sepia tone? My reference point is the perl imagemagick library, so I can easily use any basic operation. I've tried to quantize (making it grayscale), colorize, and then enhance the image but it's still a bit blurry.

11 image-processing imagemagick share improve this question follow asked Jun 29 '09 at 23:37 user83358 854 ● 3 □ 10 □ 17 add a comment

4 Answers

Active Oldest Votes

24 Sample code of a sepia converter in C# is available in my answer here: [What is wrong with this sepia tone conversion algorithm?](#)

The algorithm comes from [this page](#), each input pixel color is transformed in the following way:

```
outputRed = (inputRed * .393) + (inputGreen * .769) + (inputBlue * .189)  
outputGreen = (inputRed * .349) + (inputGreen * .686) + (inputBlue * .168)  
outputBlue = (inputRed * .272) + (inputGreen * .534) + (inputBlue * .131)
```

If any of these output values is greater than 255, you simply set it to 255. These specific values are the values for sepia tone that are recommended by Microsoft.

share improve this answer follow edited May 23 '17 at 11:54 by Community♦ answered Feb 25 '12 at 23:43 by Max Galkin 15.9k ● 9 □ 58 □ 108

You will need to use Math.Min likely. I tried doing the check for 255 after those three lines and an error will occur. I was facing the same problem earlier today when I was trying to make a sepia tone for my program.. – BigBug Feb 26 '12 at 6:34

But what if I want something different to change the filter then how can I get to these values? like my question is how we came to know about these values, do we need to just put different values again and again? – AHF Mar 23 '14 at 15:20

add a comment

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#). X

<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()

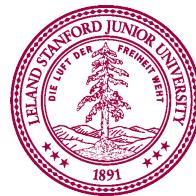
def sepia_pixel(pixel):
    R = pixel.red
    G = pixel.green
    B = pixel.blue
    pixel.red = 0.393 * R + 0.769 * G + 0.189 * B
    pixel.green = 0.349 * R + 0.686 * G + 0.168 * B
    pixel.blue = 0.272 * R + 0.534 * G + 0.131 * B
```



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():

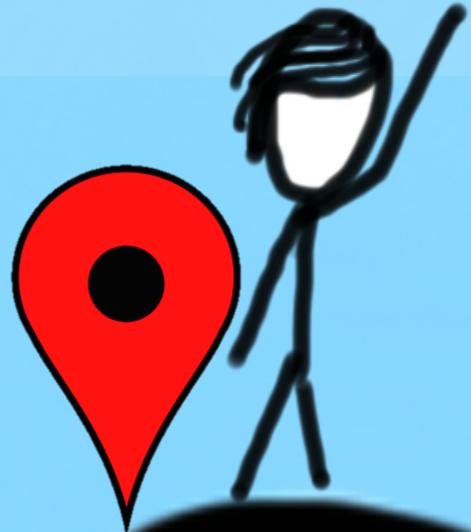
    for y in range(600):
        for x in range(800):
            print(x, y)
```



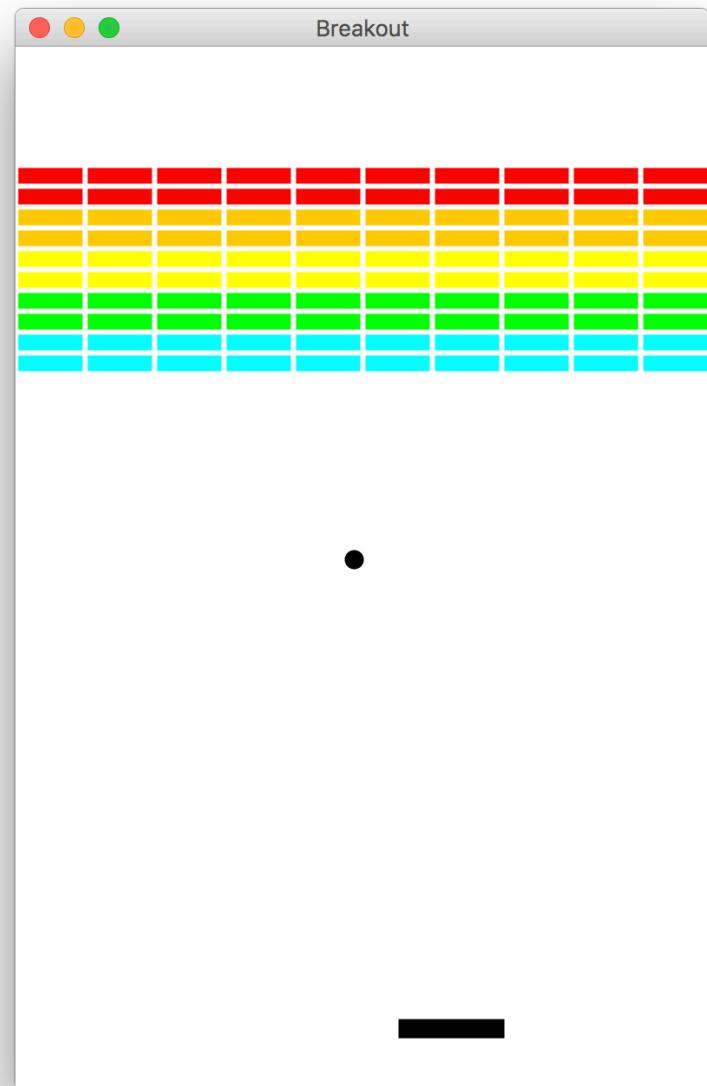
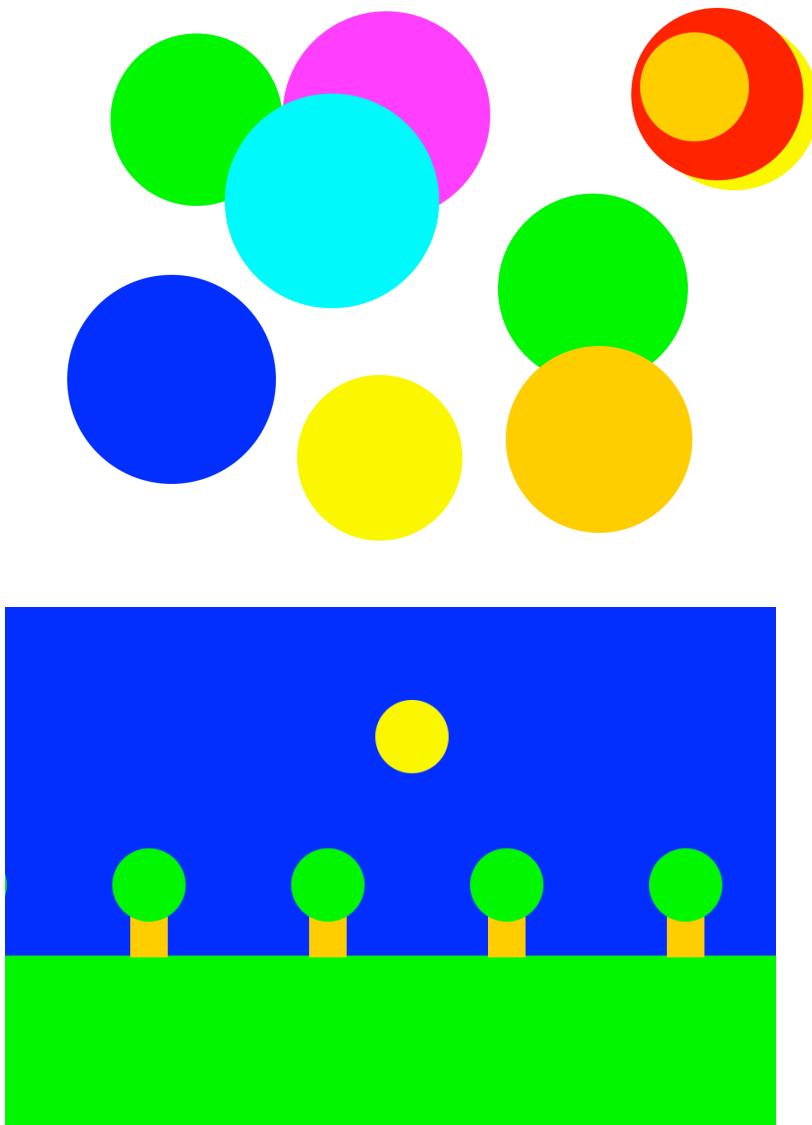
End Review

Today's Goal

1. How do I draw shapes?



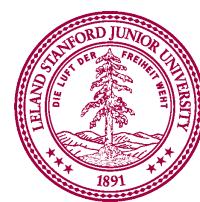
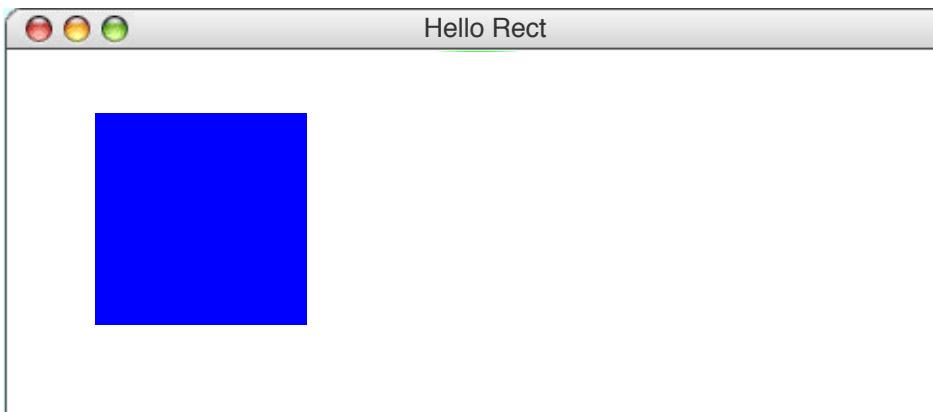
Graphics Programs



Draw a Rectangle

the following `main` method displays a blue square

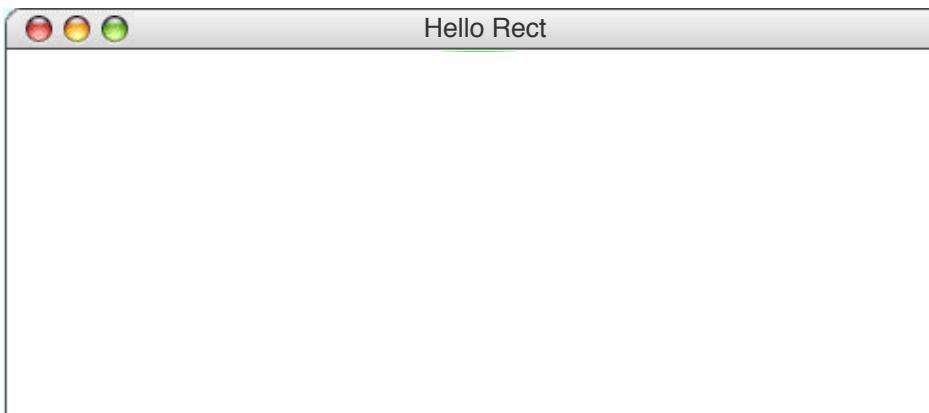
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
    canvas.mainloop()
```



Draw a Rectangle

the following `main` method displays a blue square

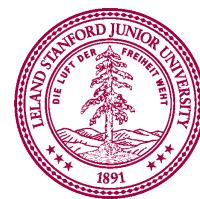
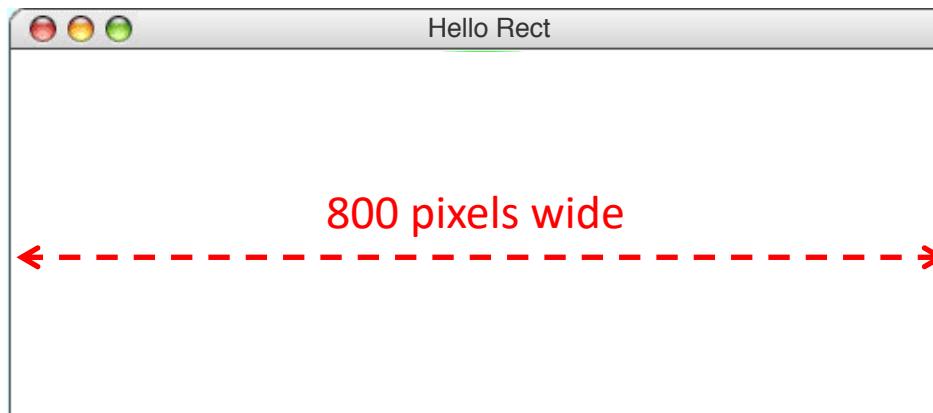
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

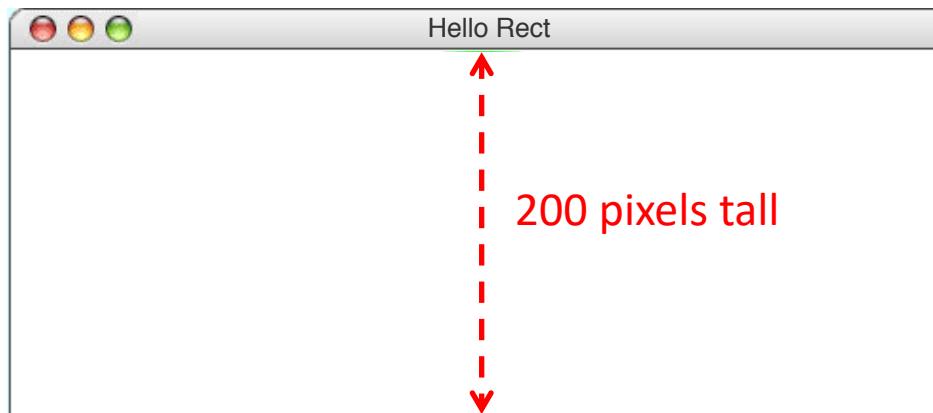
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



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

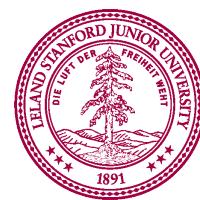
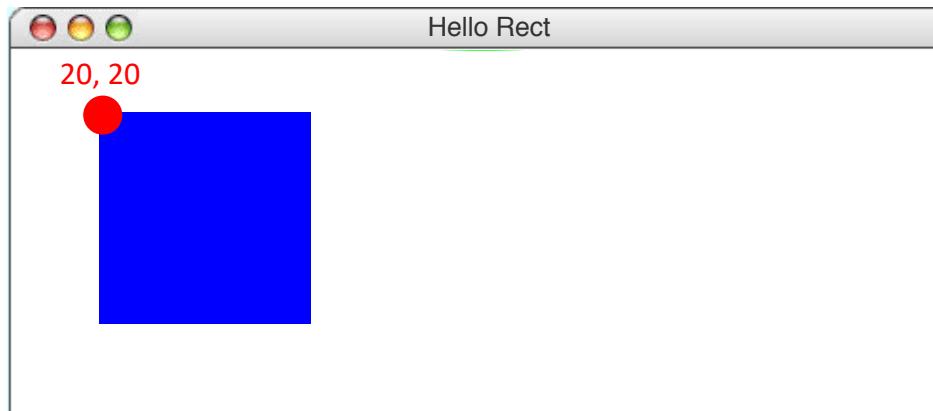
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

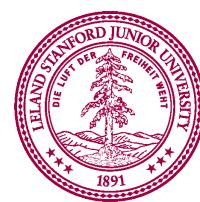
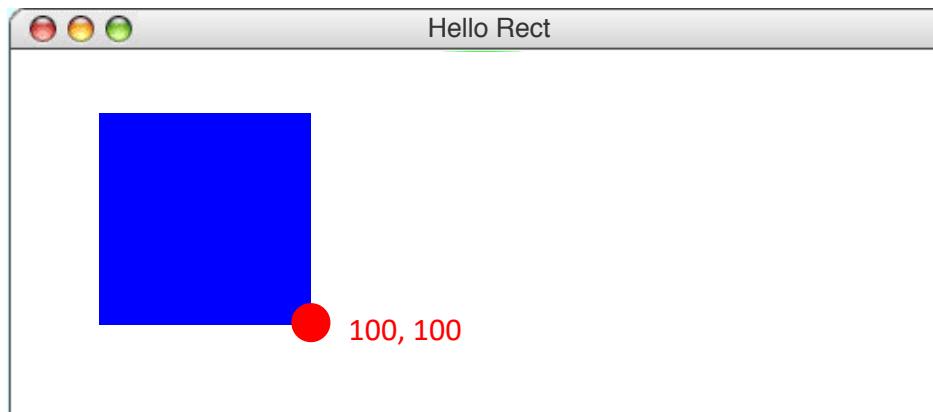
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

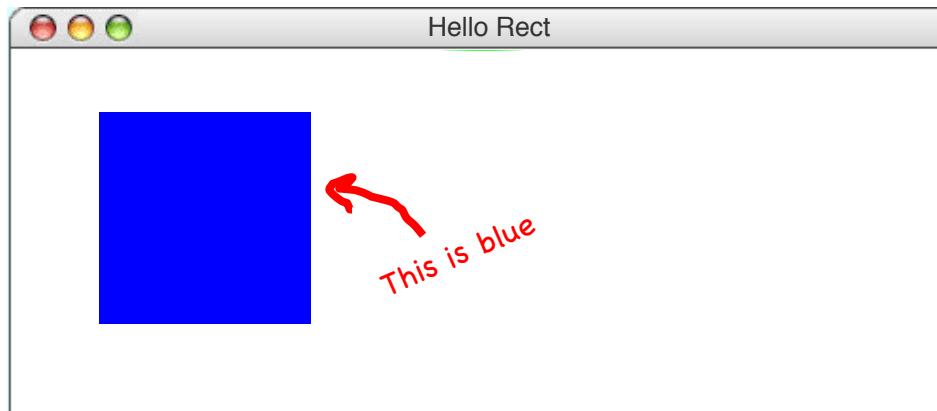
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



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")
```



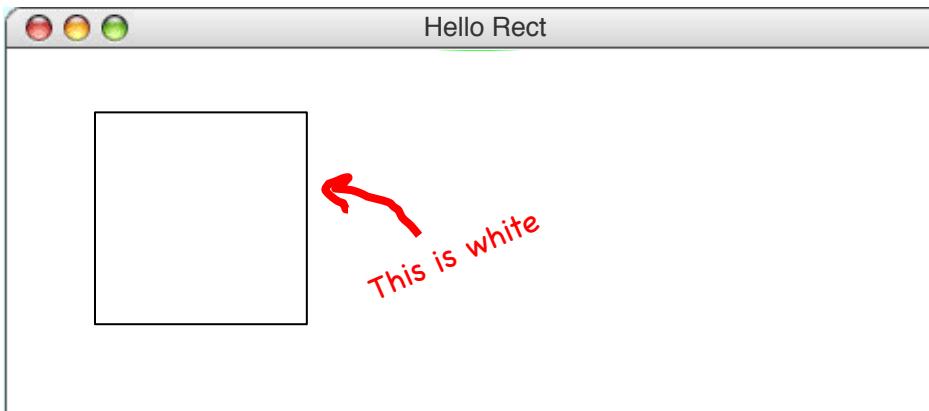
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



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)
```



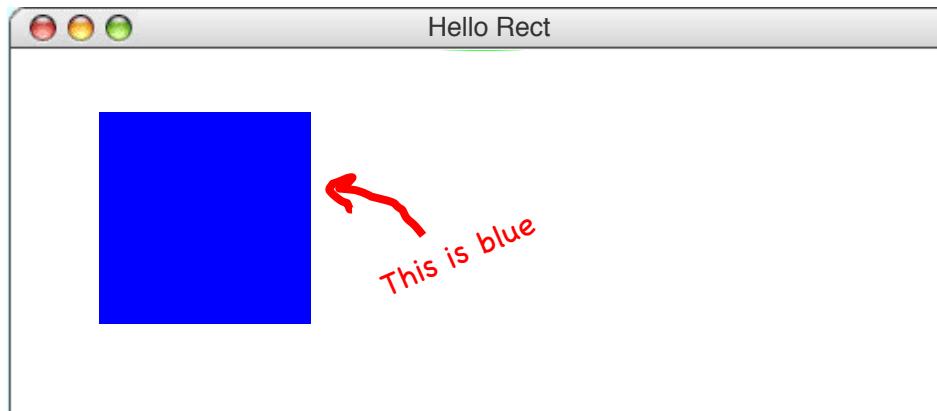
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



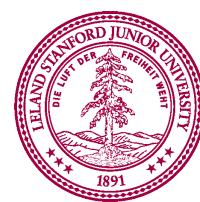
Draw a Rectangle

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")
```



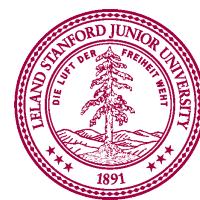
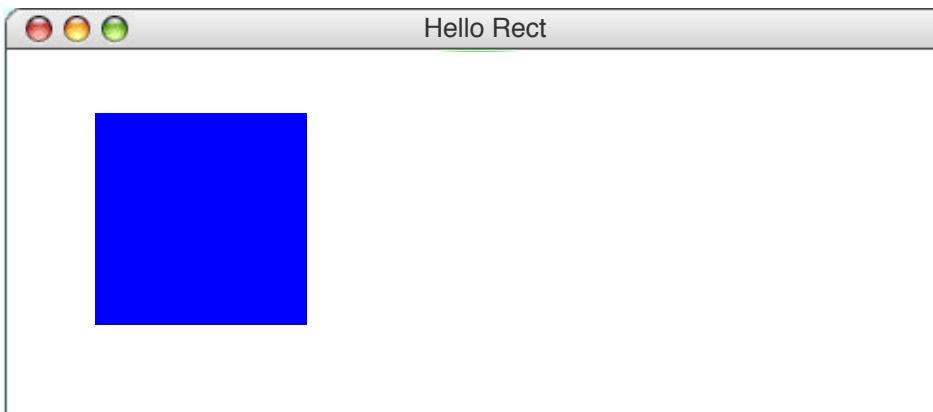
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



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

0,0

x 40,20

x 120,40

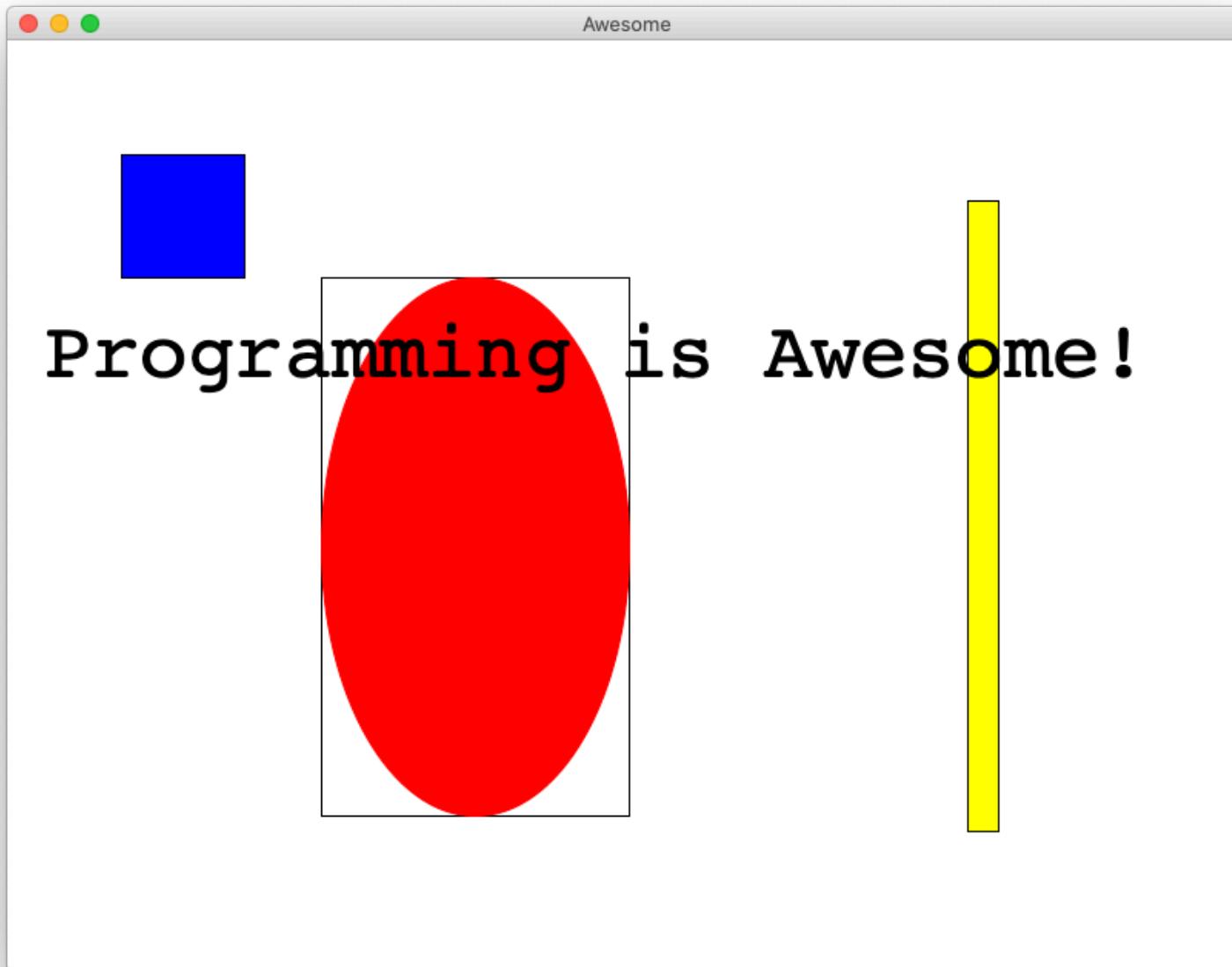
x 40,120

CANVAS_WIDTH

CANVAS_HEIGHT



Rectangles, Ovals, Text



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

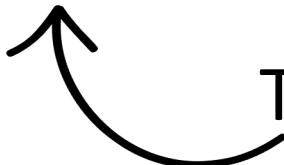


- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`



- `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)**

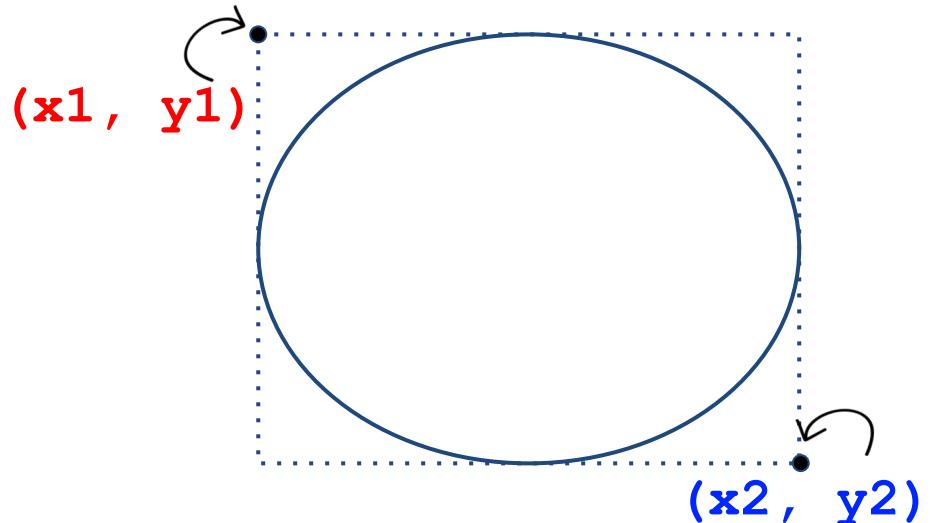
- **canvas.create_oval()**
- **canvas.create_text()**



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



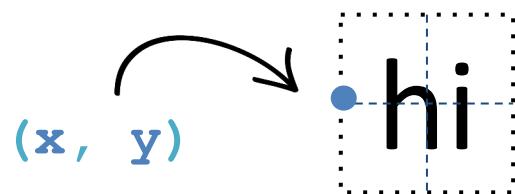
- `canvas.create_line()`
- `canvas.create_oval(x1, y1, x2, y2)`
- `canvas.create_text()`



- `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')`



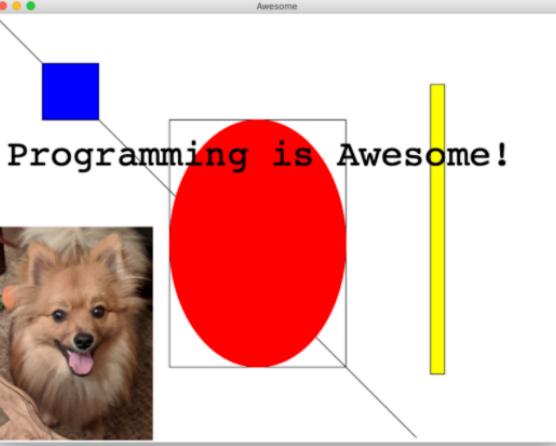
Pedagogy

CS106A Lectures Assignments Section Handouts Examples Schedule

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.



```
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/
```

Handouts Examples

General Information

Course Placement

Honor Code

Installing PyCharm

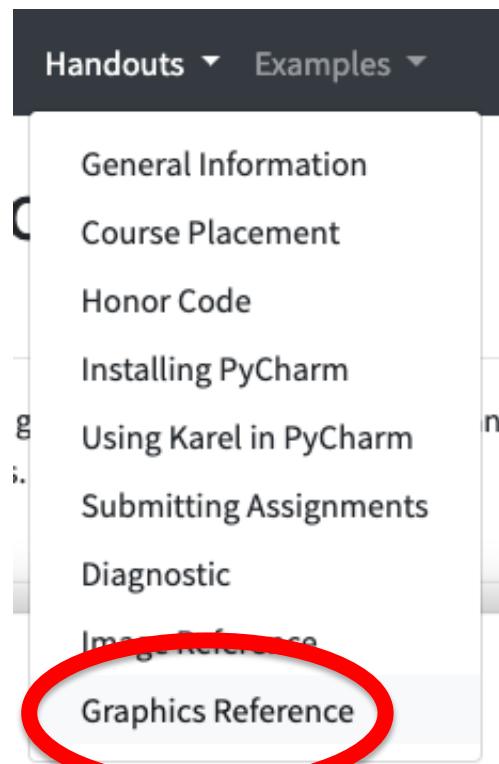
Using Karel in PyCharm

Submitting Assignments

Diagnostic

Image Reference

Graphics Reference



Solution

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

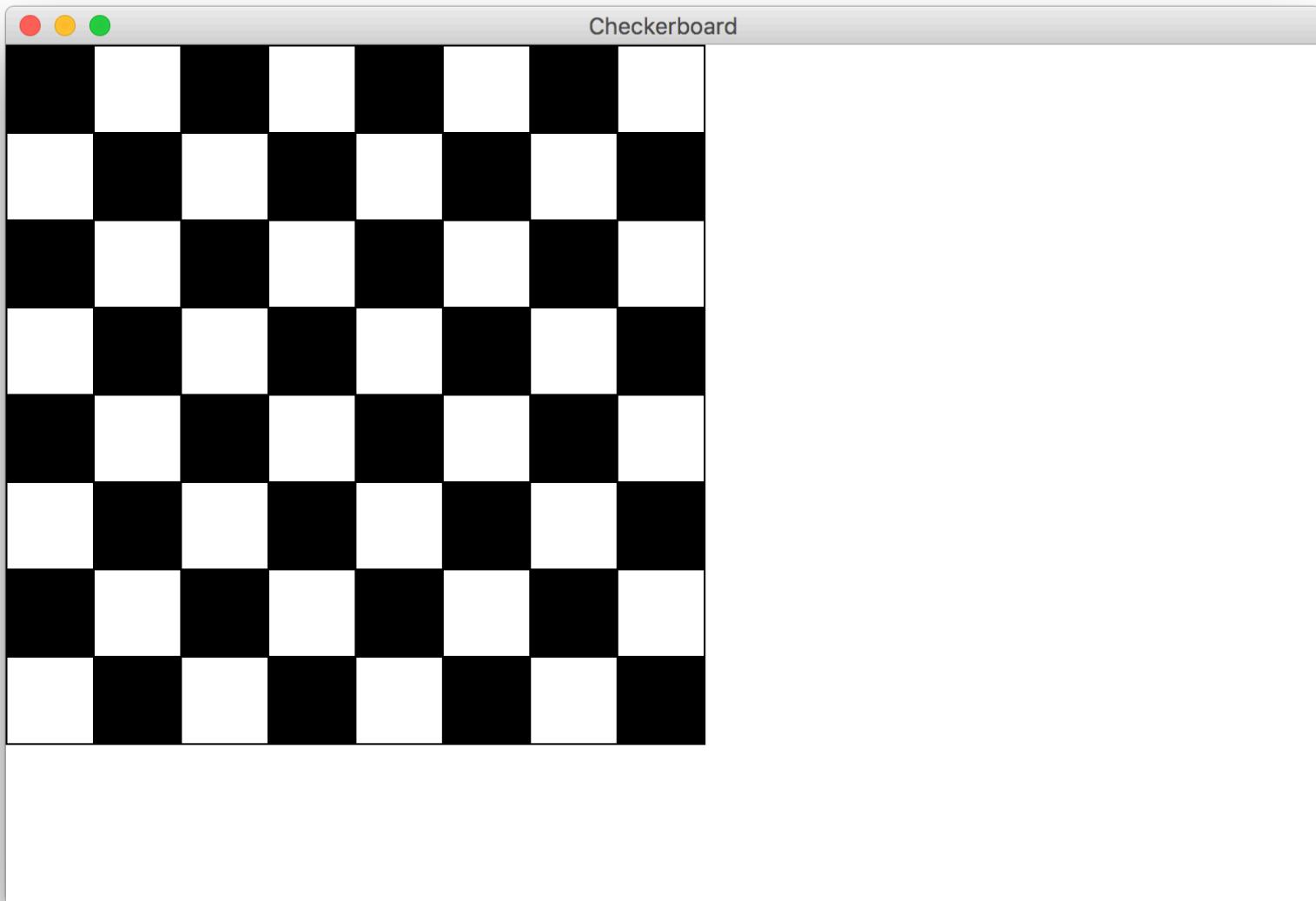
CANVAS_WIDTH = 800
CANVAS_HEIGHT = 600

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

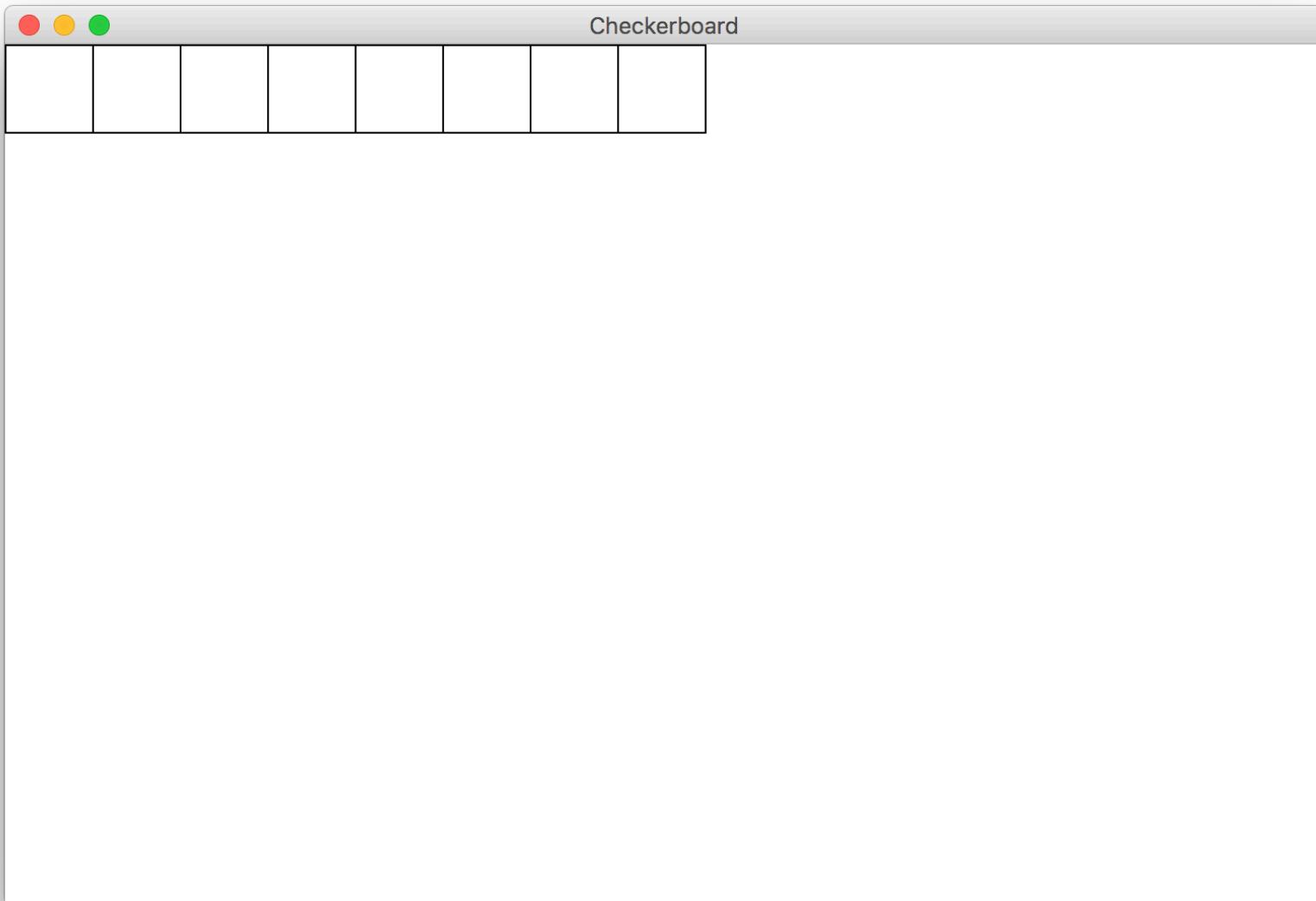
    # a blue square with width and height = 80
    canvas.create_rectangle(70, 70, 150, 150, fill="blue")
    # a yellow rectangle that is long and skinny
    canvas.create_rectangle(620, 100, 640, 510, fill="yellow")
```



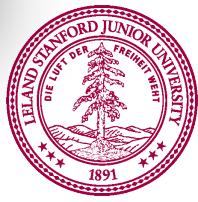
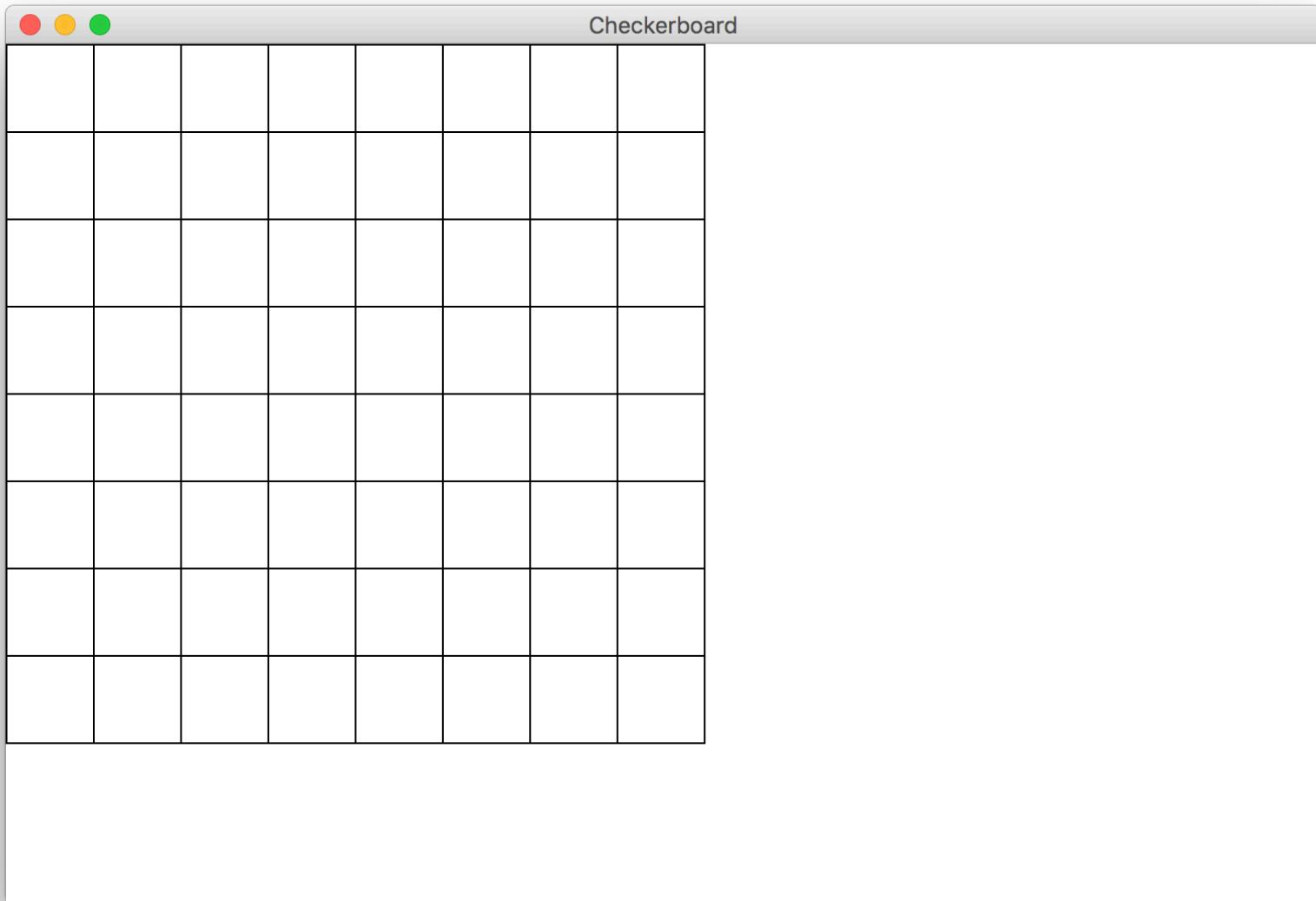
Goal



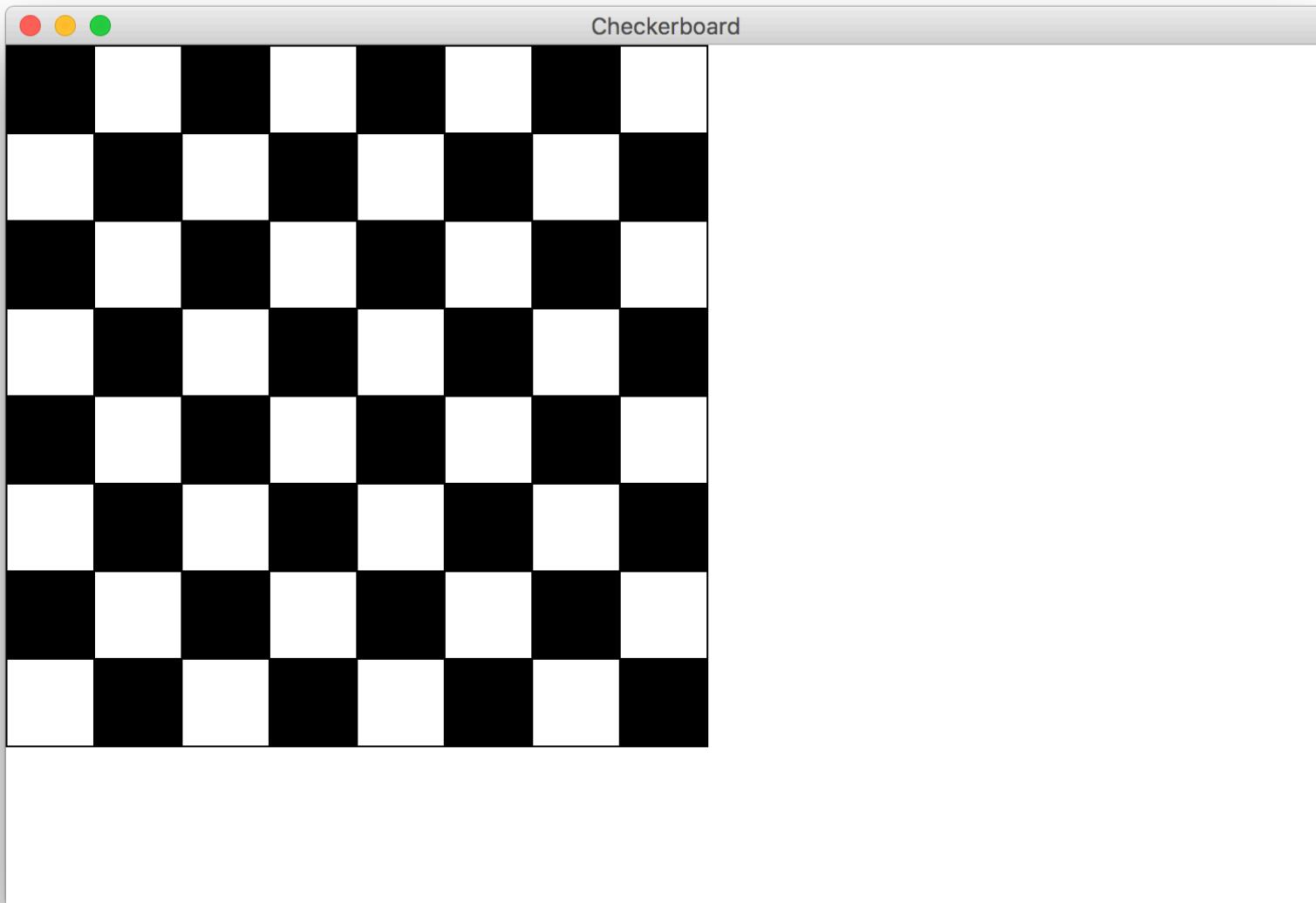
Milestone 1

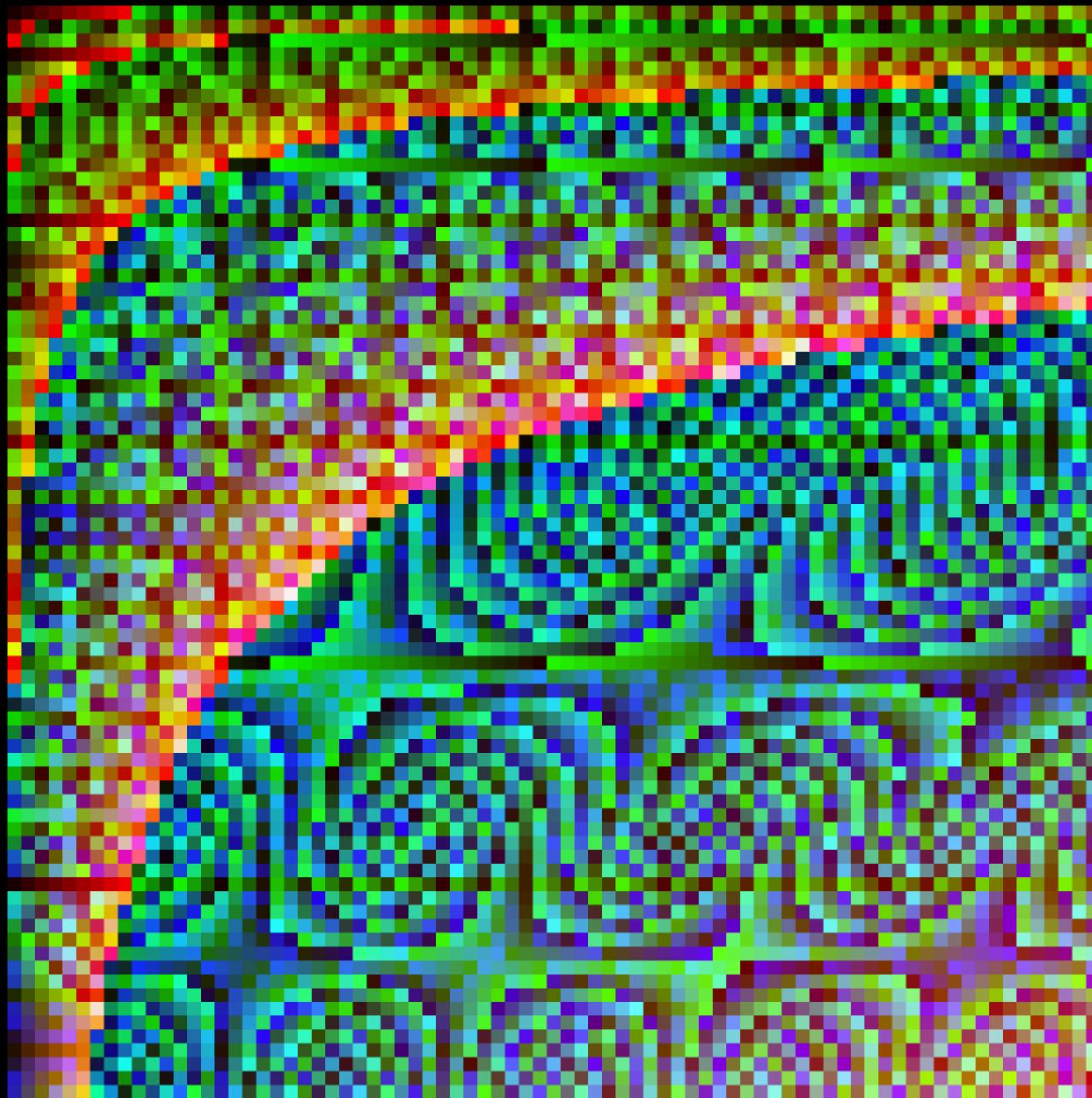


Milestone 2



Milestone 3





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?

The screenshot shows a Google Docs interface with the following details:

- Title:** Apollo 11 research
- Toolbar:** Includes File, Edit, View, Insert, Format, Tools, Table, Add-ons, Help, and a note that the last edit was 3 hours ago.
- Document Outline:** On the left, it lists sections: Apollo 11 (selected), Summary, The Spacecraft, Design, Command module, Service module, The People, Neil Armstrong, Buzz Aldrin, Mission Highlights, The Launch, The Landing, and Return Trip.
- Main Content Area:**
 - Apollo 11:** The title is bolded.
 - Summary:** A section header. The text describes the Apollo 11 moon mission where Neil Armstrong, Buzz Aldrin, and Michael Collins landed at Tranquility Base on the moon on July 20, 1969.
 - The Spacecraft:** A section header. The text states that the Apollo 11 mission had three spacecraft: the Command Module Columbia, a Service Module, and the Lunar Module Eagle. Columbia was the only part of the spacecraft to return to Earth.
 - Design:** A section header. The text mentions the key NASA spacecraft involved in the Apollo 11 mission: a Saturn V rocket, an Apollo CSM-107 (Command/Service Module), and an Apollo LM-5 (Lunar Module, AKA "The Eagle").
 - Command module:** A section header. The text describes the Command/Service Module (CSM) as one of two spacecraft, along with the Lunar Module, used for the United States Apollo program.
 - Service module:** A section header. The text explains that the Service Module contained oxygen, water, and electric power for the command module, and housed the service propulsion system.

<https://docs.google.com/document/d/1eBtnEiiI3KHe fFS-kSAOpXqeSXpbfTTMlmOgj6I9dvk/>



```
def main():
```

```
    canvas = make_canvas(...)
```

```
    draw_square(canvas)
```

```
def draw_square(canvas):
```

```
    canvas.create_rectangle(20, 20, 100, 100)
```

stack

heap

```
main
```



```
def main():
```

```
    canvas = make_canvas(...)
```

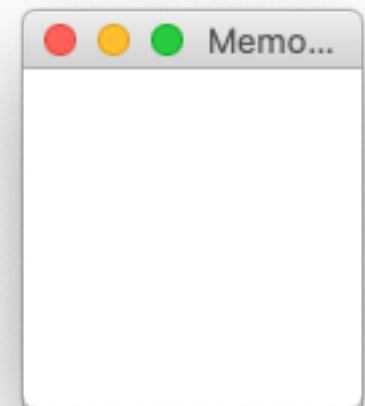
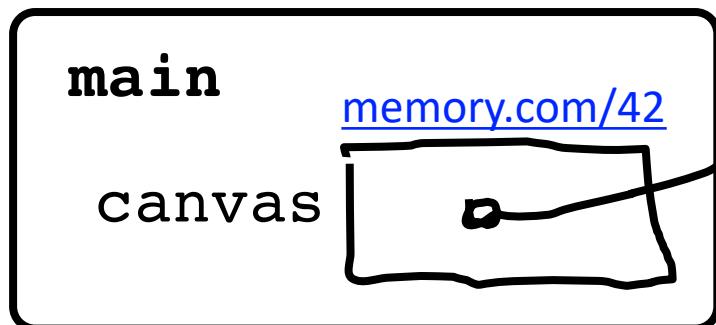
```
    draw_square(canvas)
```

```
def draw_square(canvas):
```

```
    canvas.create_rectangle(20, 20, 100, 100)
```

stack

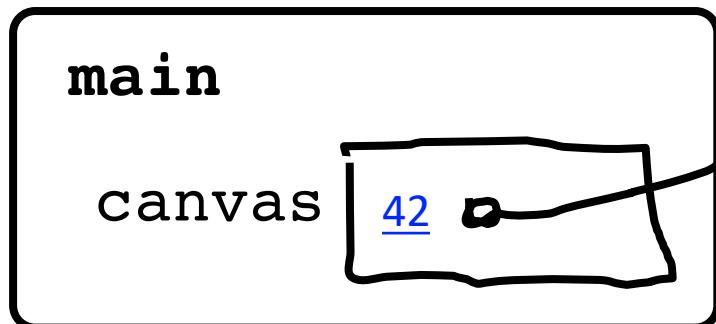
heap



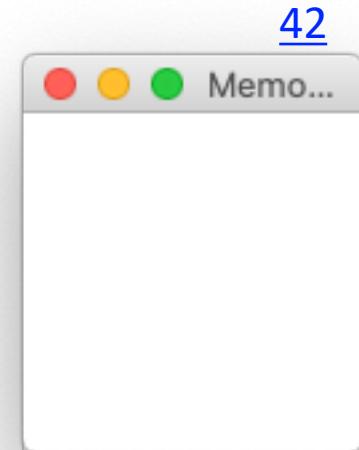
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

stack



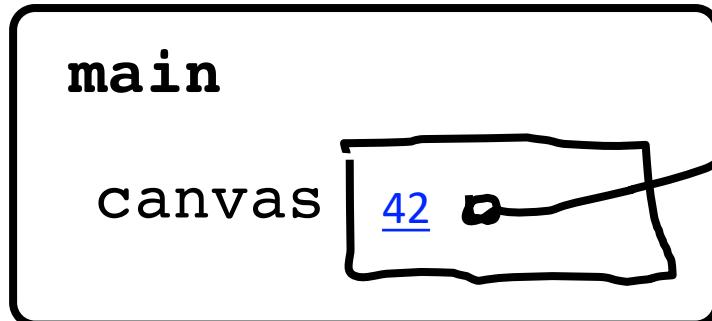
heap



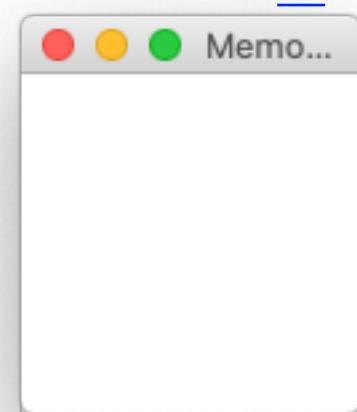
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

stack

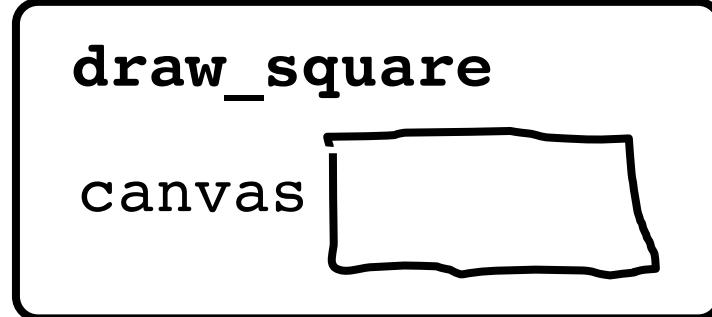


heap



42

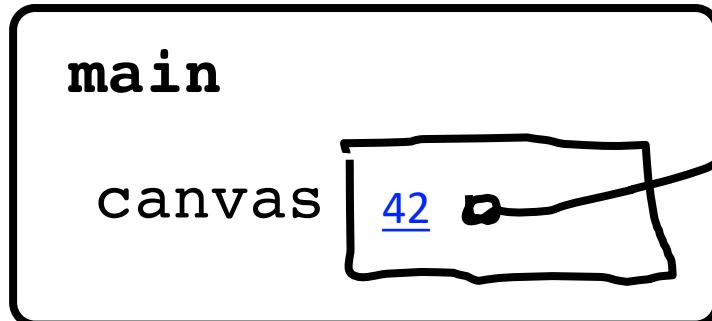
draw_square



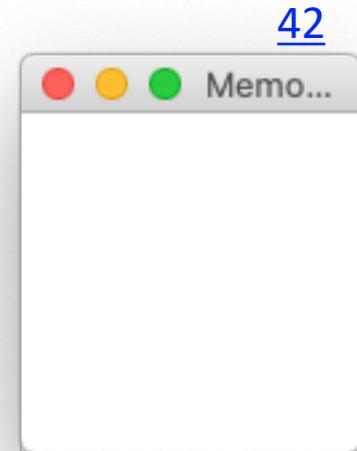
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

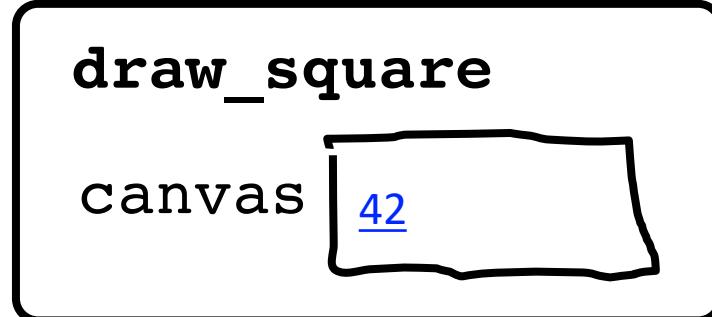
stack



heap



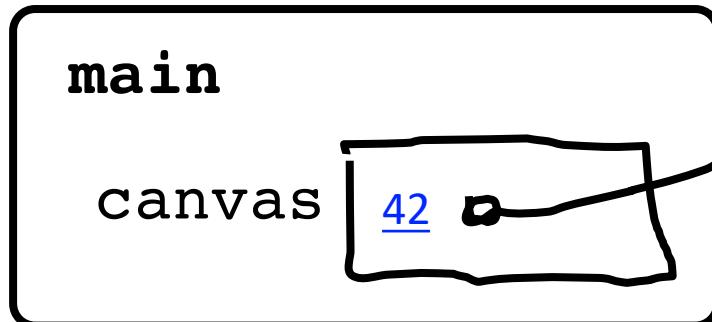
draw_square



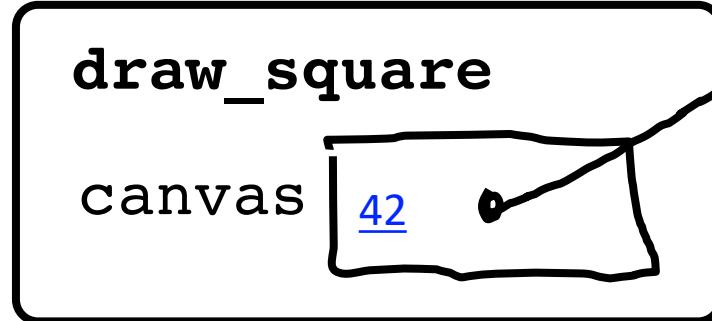
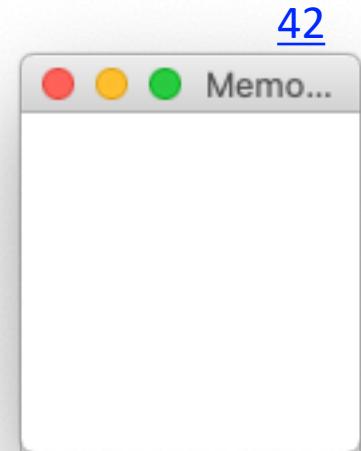
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

stack



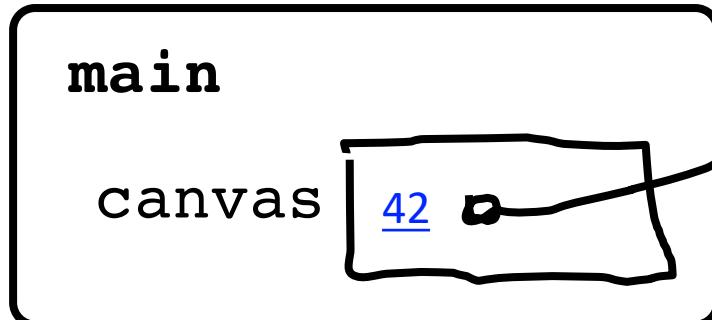
heap



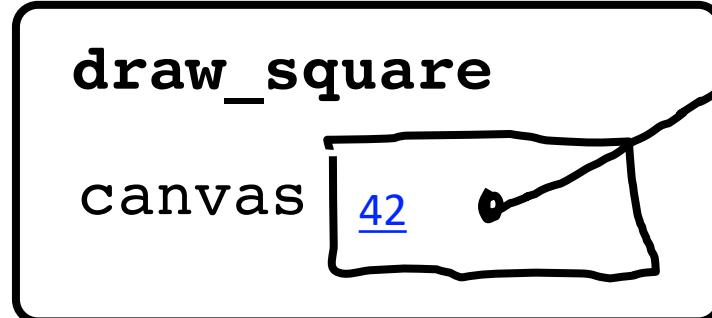
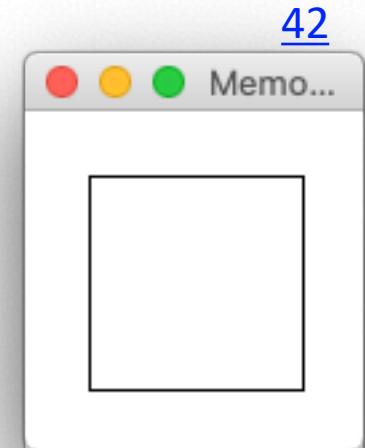
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

stack



heap

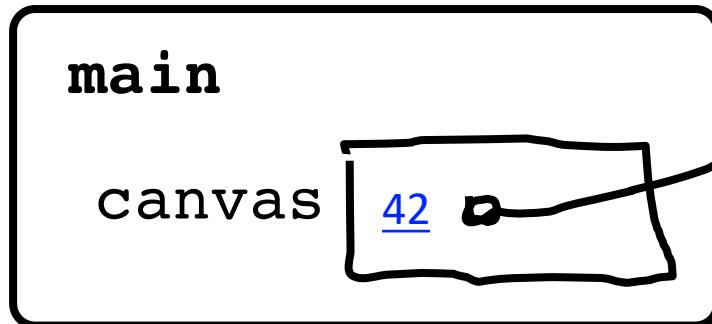


```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

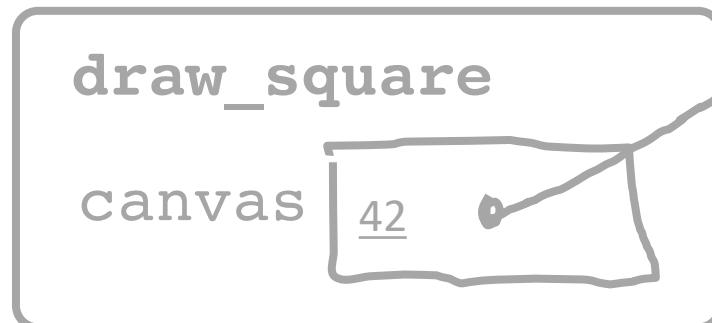
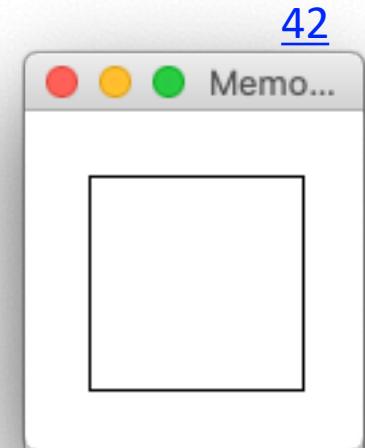
```
def draw_square(canvas):
    canvas.create_rectangle(20, 20, 100, 100)
```



stack



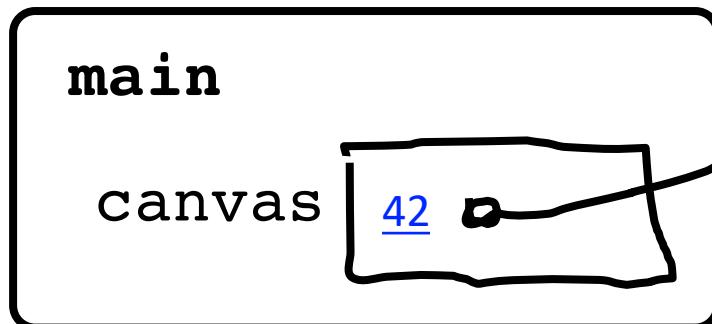
heap



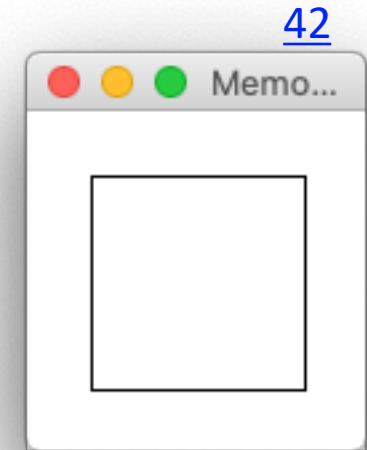
```
def main():
    canvas = make_canvas(...)
    draw_square(canvas)
```

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

stack



heap



42





Large variable types are
stored as memory
addresses

(which are like memory
URLs)

