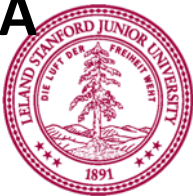
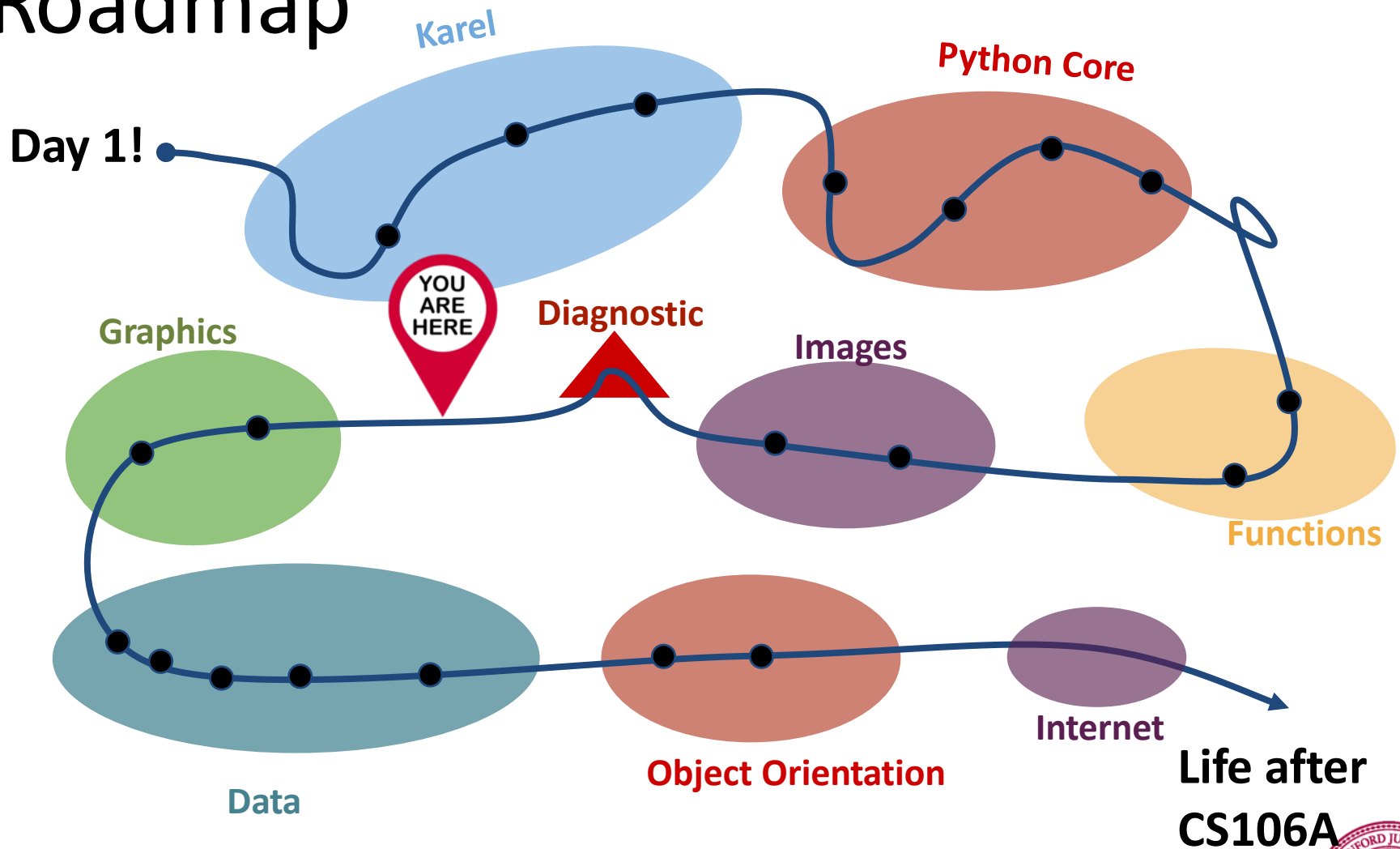




Graphics

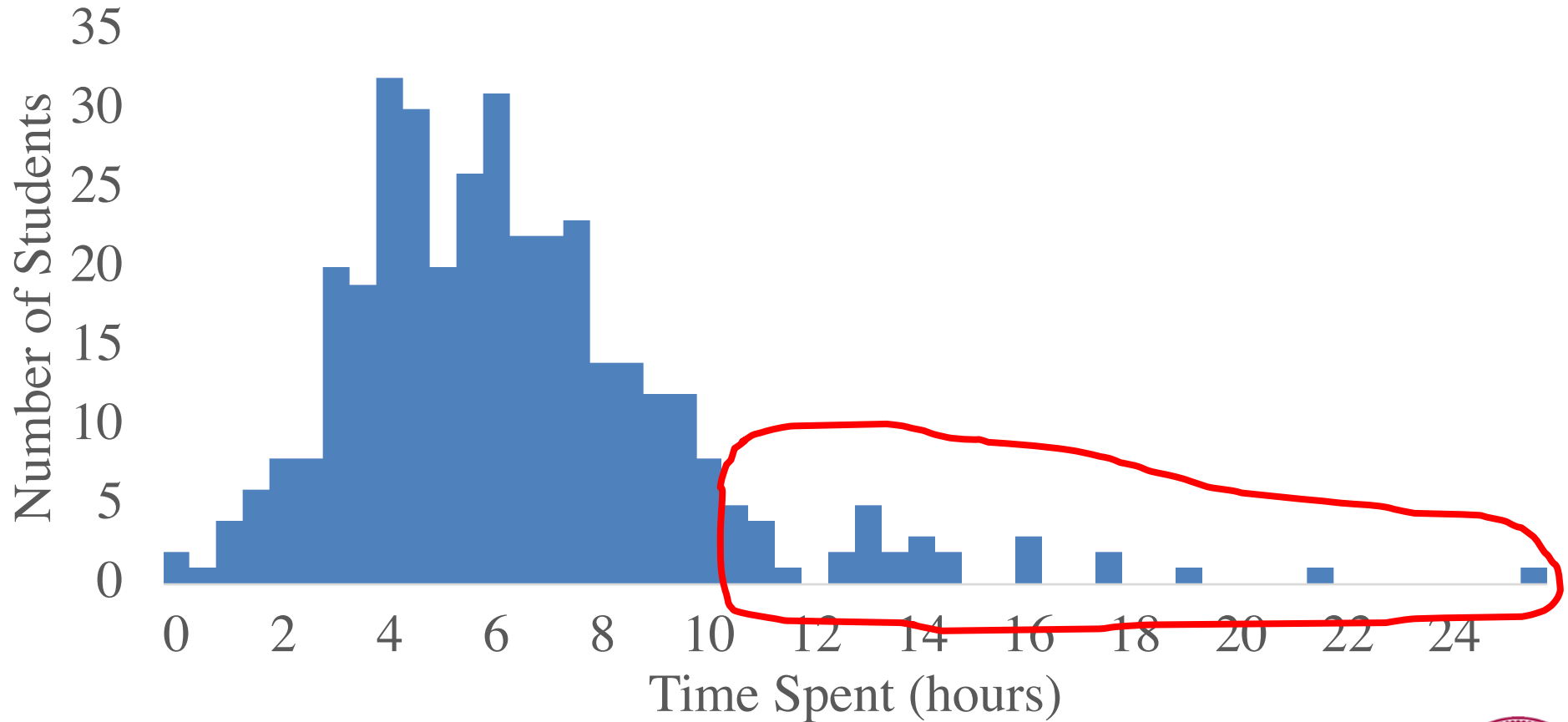
Chris Piech and Mehran Sahami
CS106A, Stanford University

Roadmap

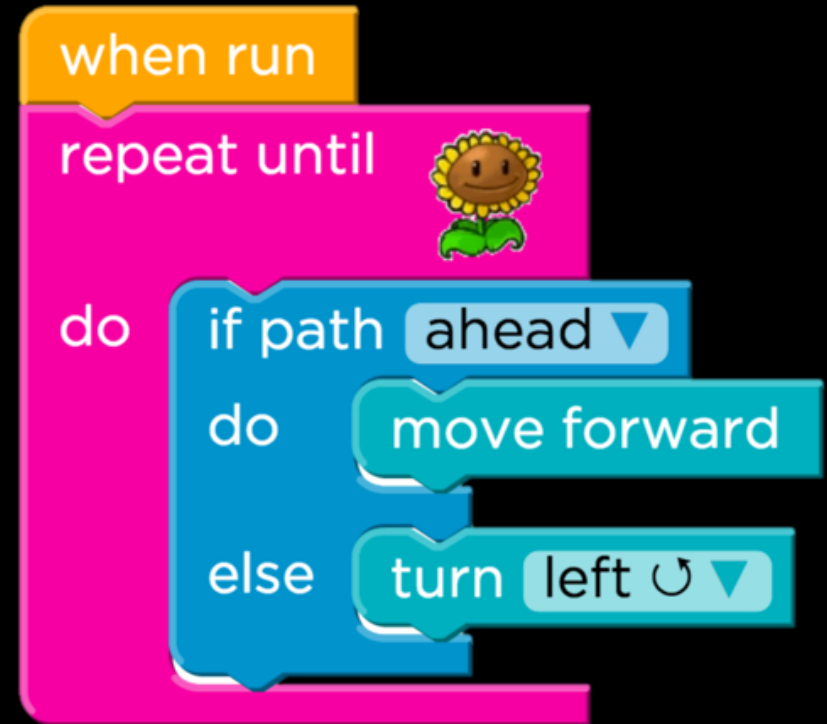


Learn by Doing

Assignment 2



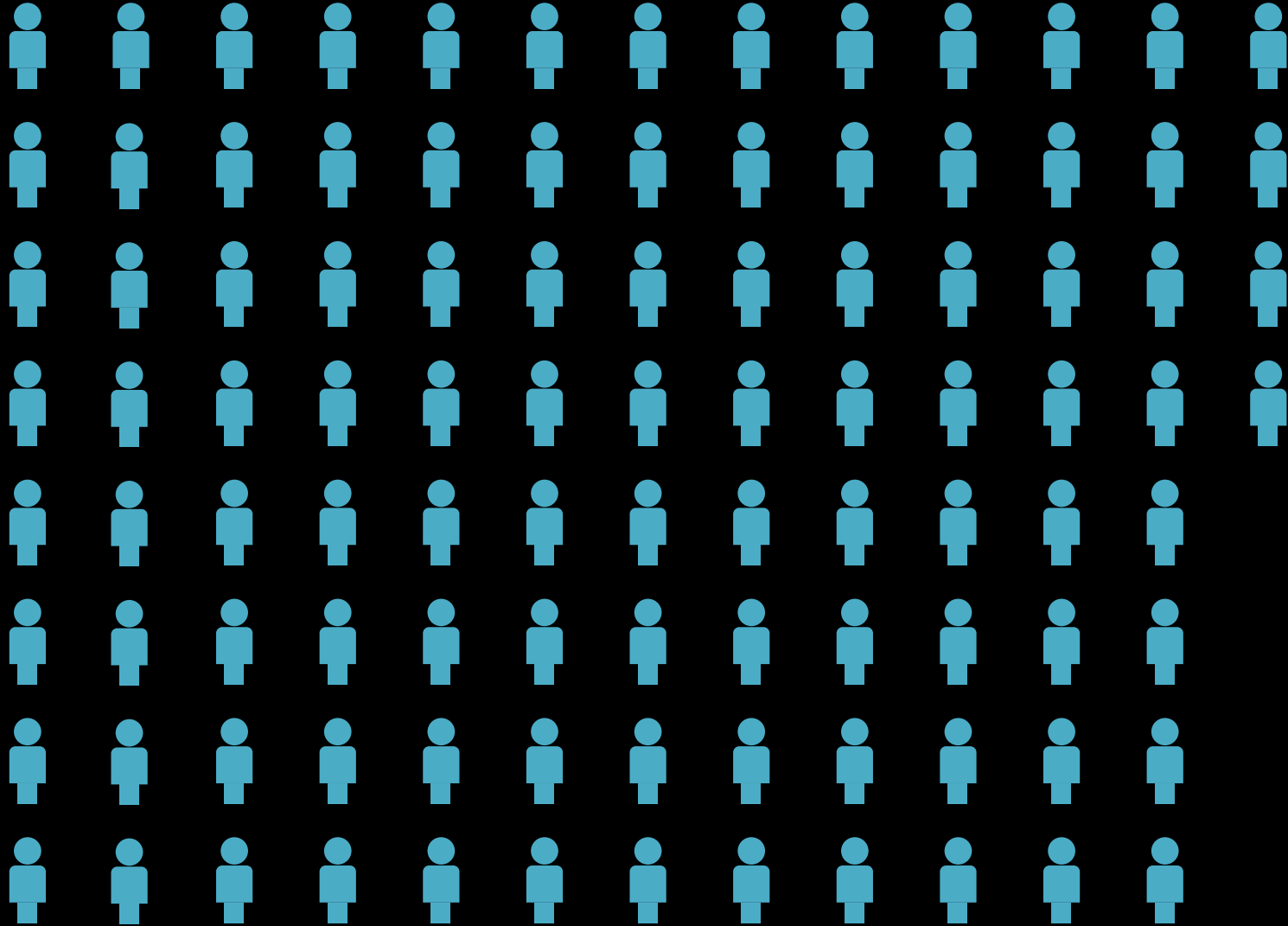
Learning to Program on the Internet



Task

Almost a hundred thousand
unique solutions

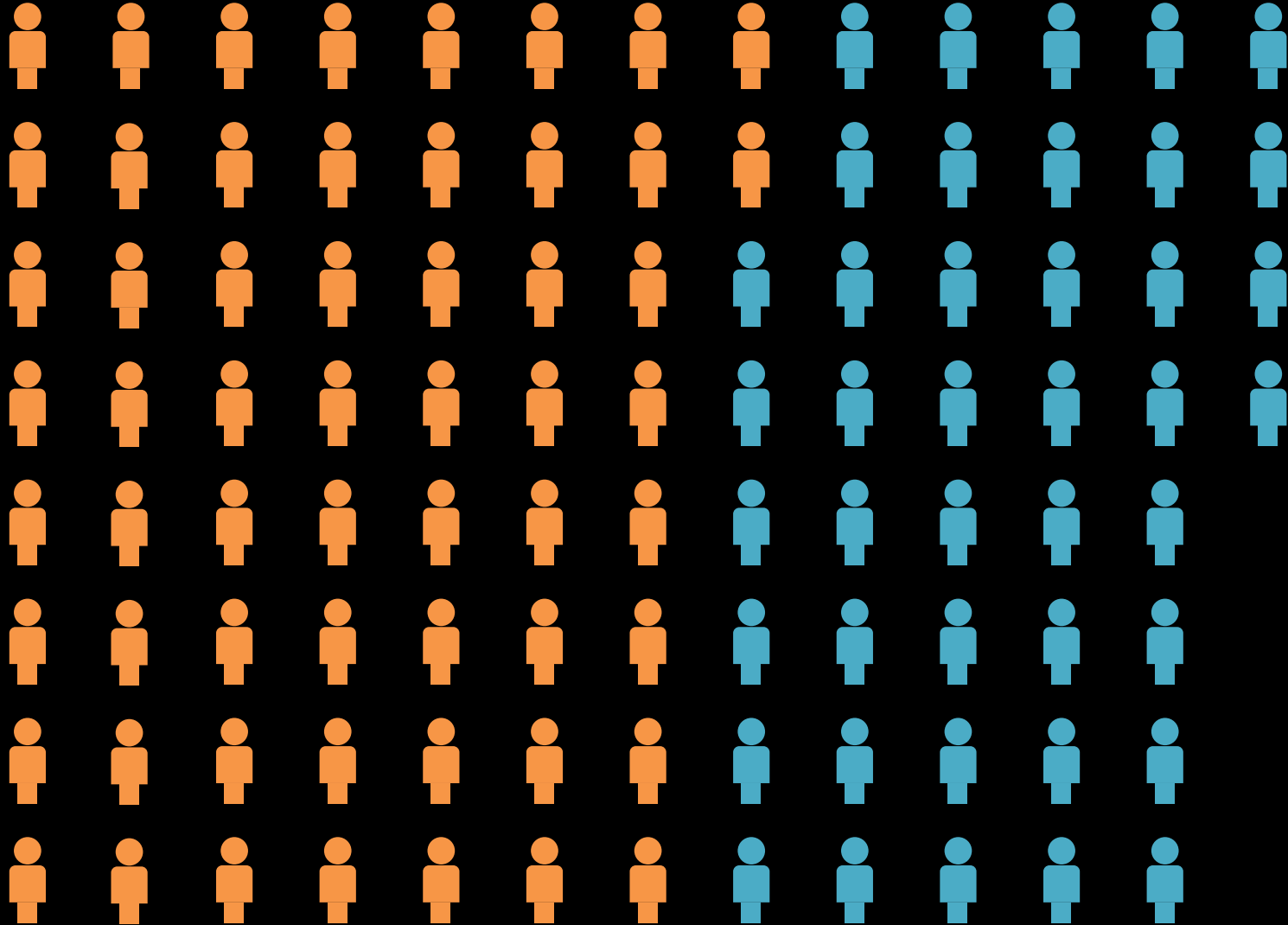
US K-12 Students



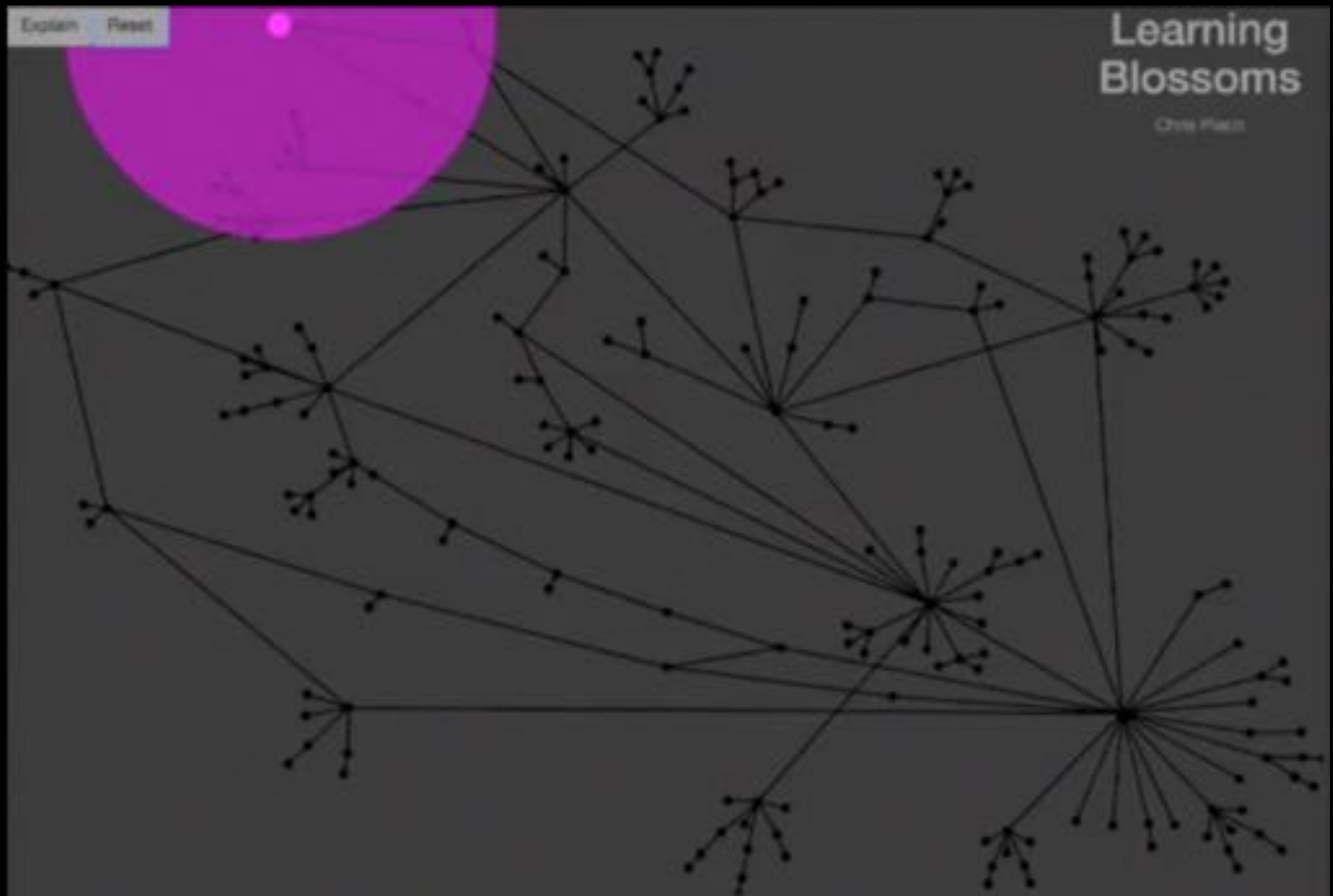
= 500,000 learners



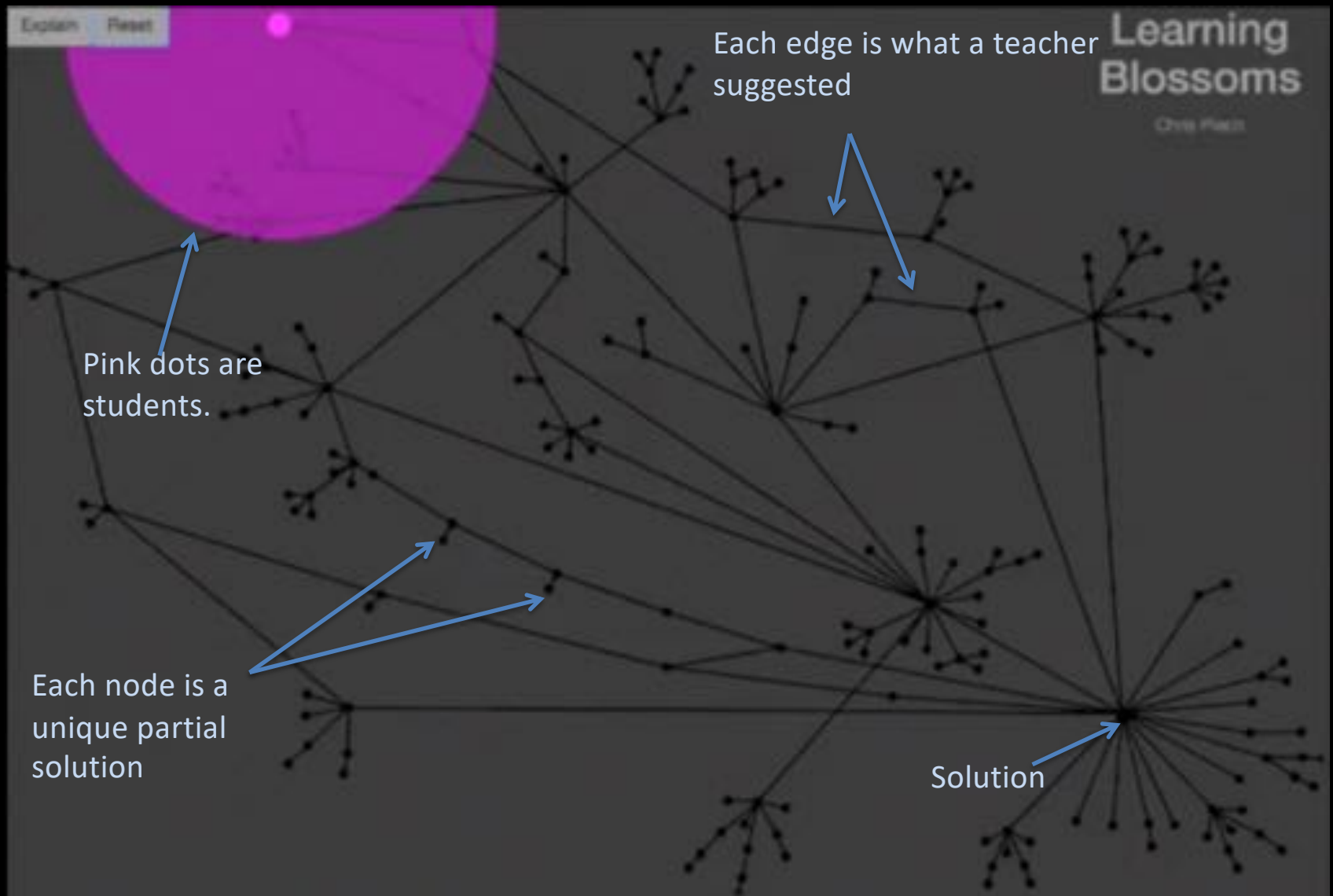
Code.org Students



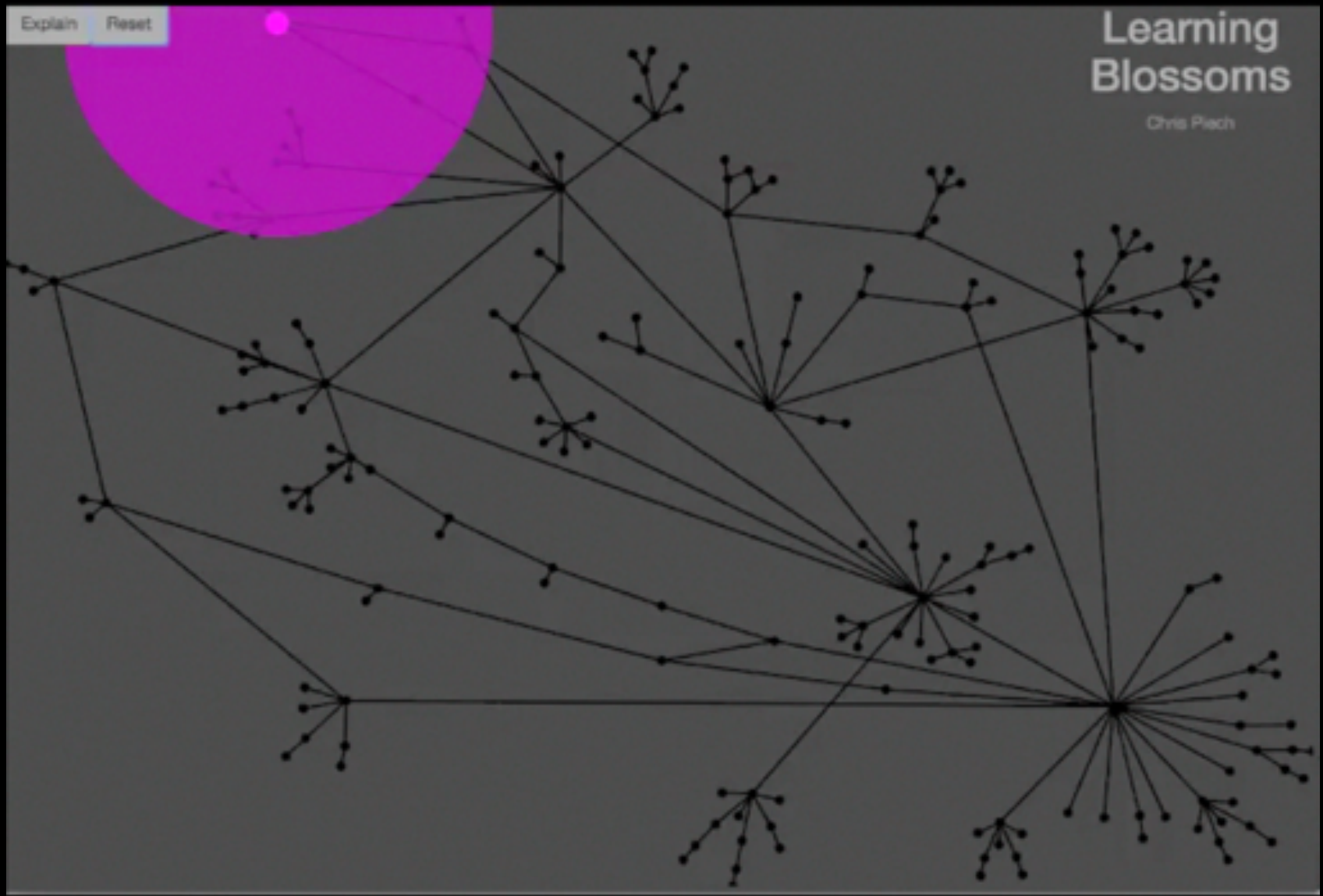
= 500,000 learners



Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.

Desirable Path Algorithm

Poisson Common Path

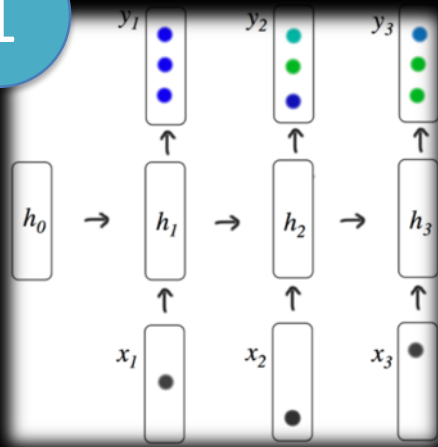
$$\gamma(s) = \arg \min_{p \in Z(s)} \sum_{x \in p} \frac{1}{\lambda_x}$$

Diagram illustrating the Poisson Common Path algorithm formula:

- $\gamma(s)$: Predicted next partial solution
- $p \in Z(s)$: Paths to solution
- $\sum_{x \in p}$: Partial solutions in the path
- $\frac{1}{\lambda_x}$: Submission count of partial solution (labeled as Path Cost)

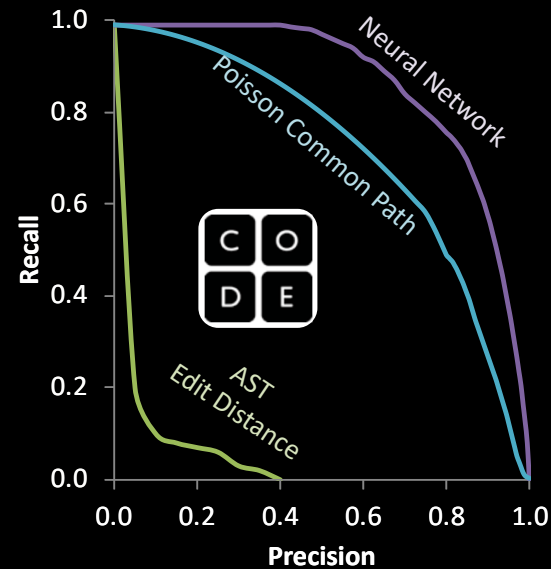
Deep Learning Algorithms

1

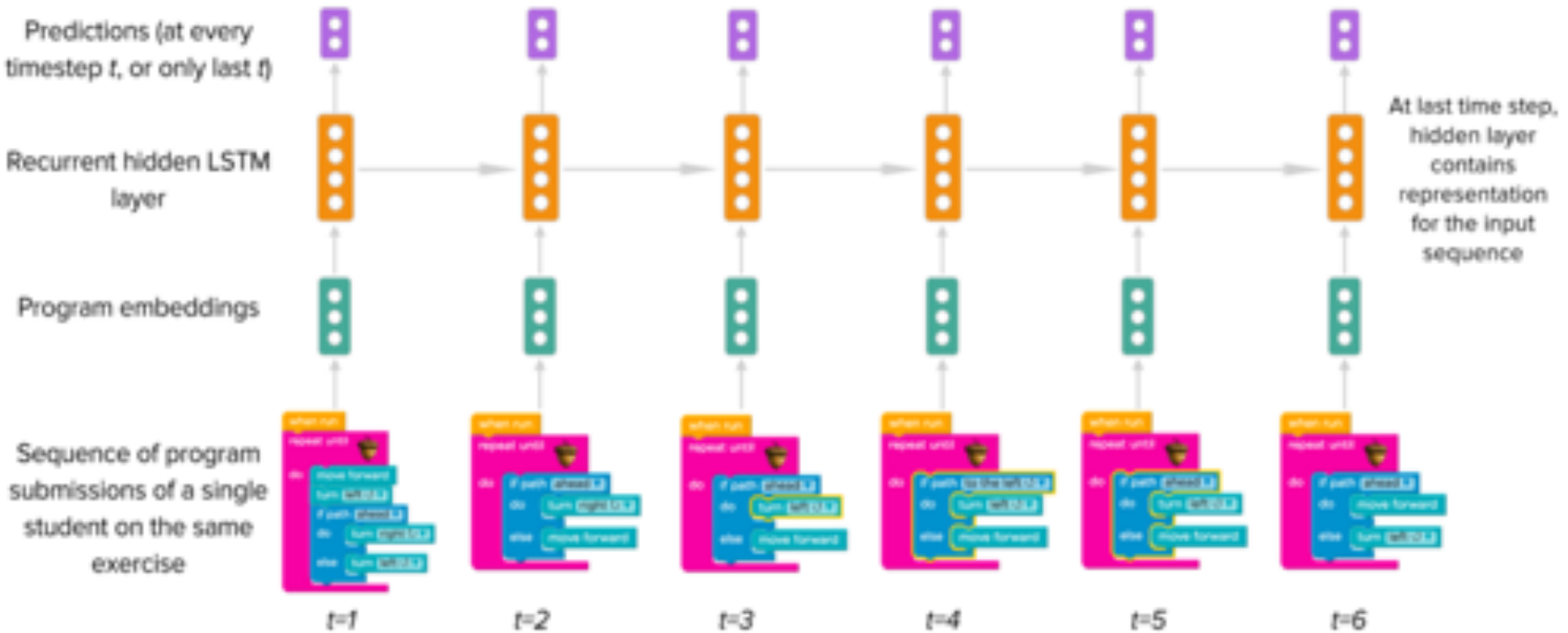


2

Program $\rightarrow \mathbb{R}^n$



Deep Learning on Trajectories

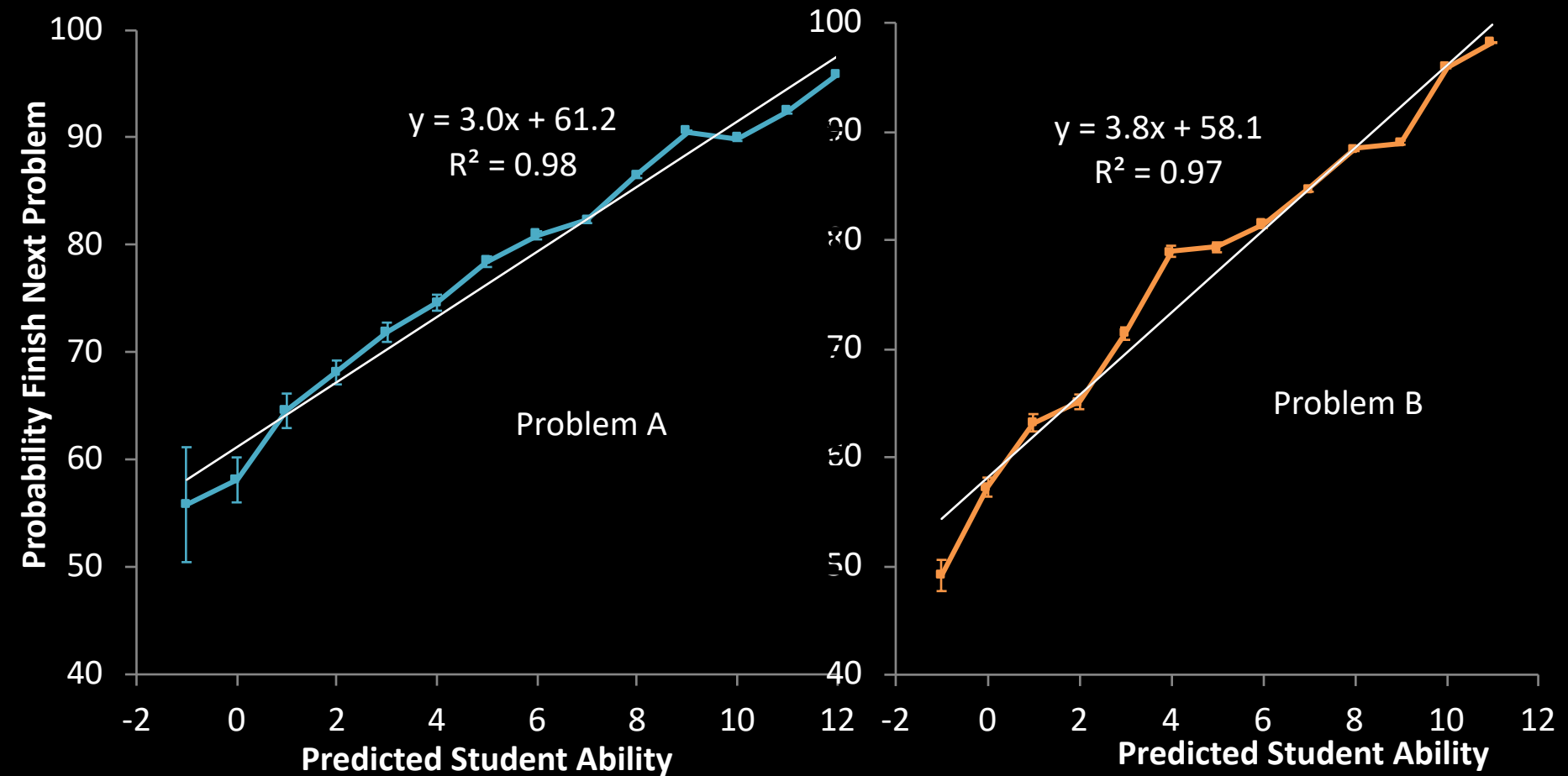


Research in collaboration with Lisa Wang

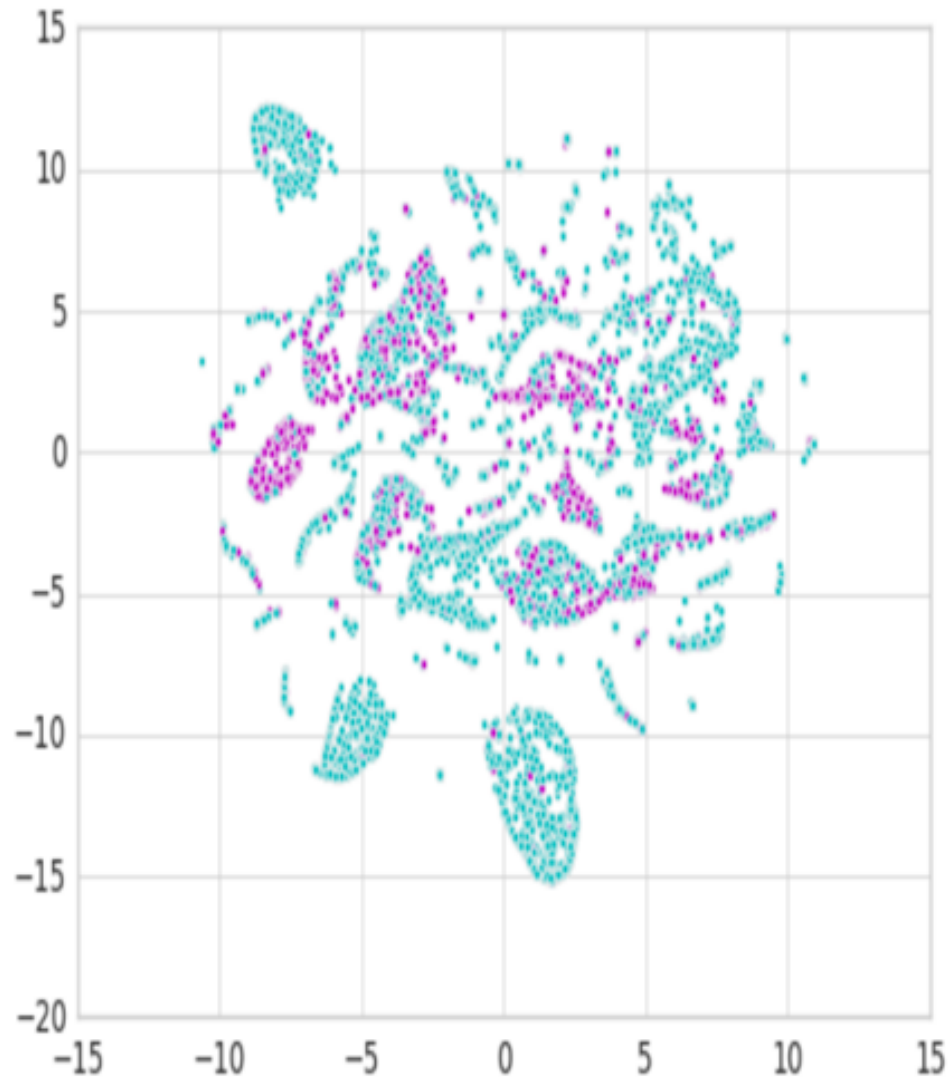
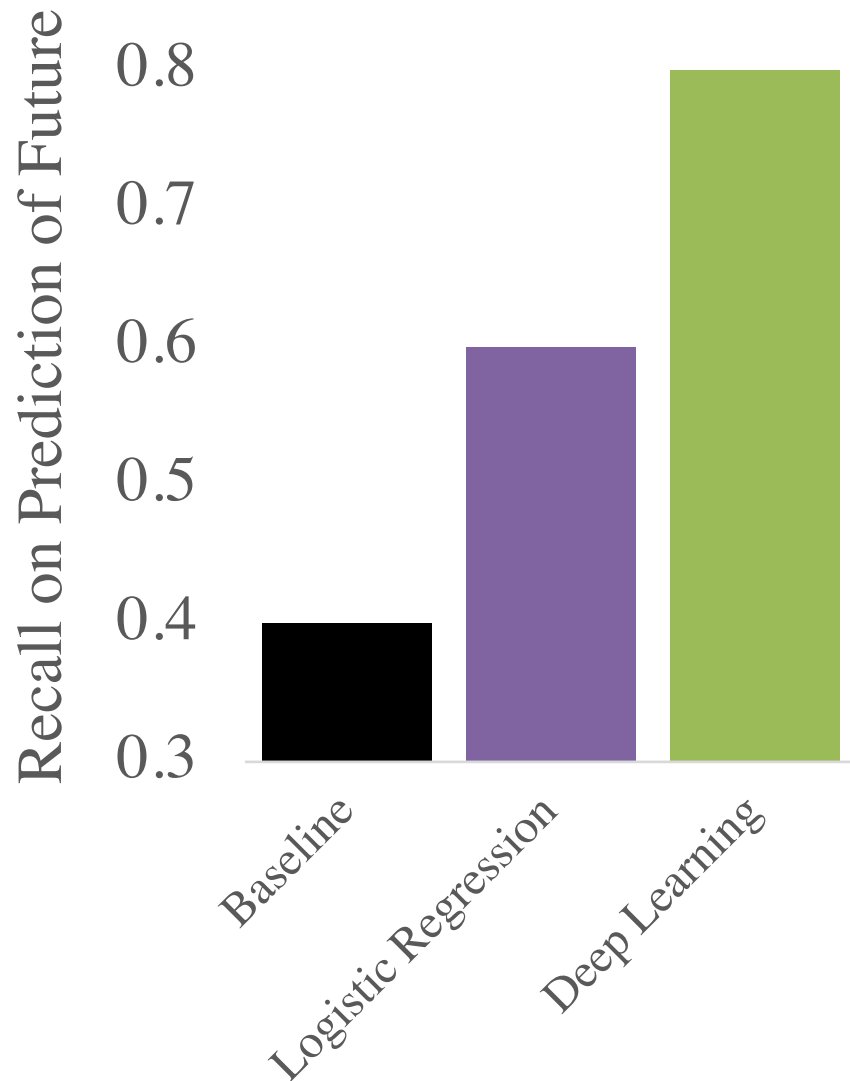
Piech + Sahami, CS106A, Stanford University



Predicts Future Success

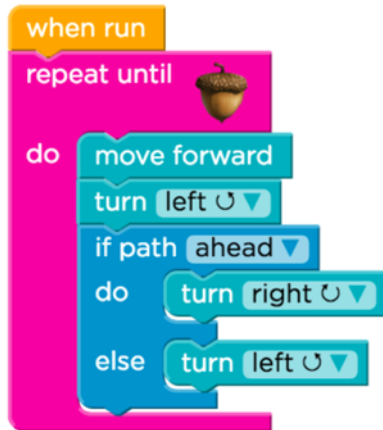


Predicts Future

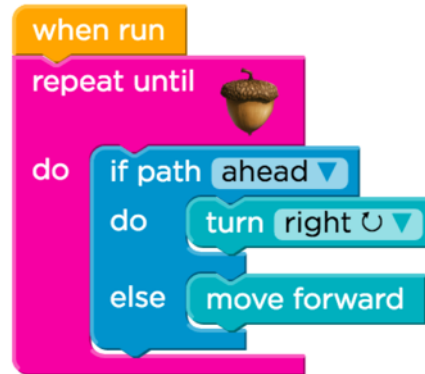


Highly Rates Grit

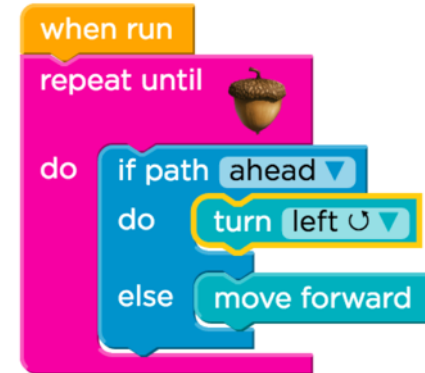
1. Two compound errors



2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



6. Finds solution



Review





Image processing - How is a sepia tone created

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

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

How is a sepia tone created?

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

11

image-processing image-magick

asked Jun 29 '09 at 23:37
user33358
854 ● 3 ● 10 ● 17

4 Answers

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.

edited May 23 '17 at 11:54
Community ●
1 ● 1

answered Feb 25 '12 at 23:43
Max Galkin
15.9k ● 5 ● 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

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



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
```



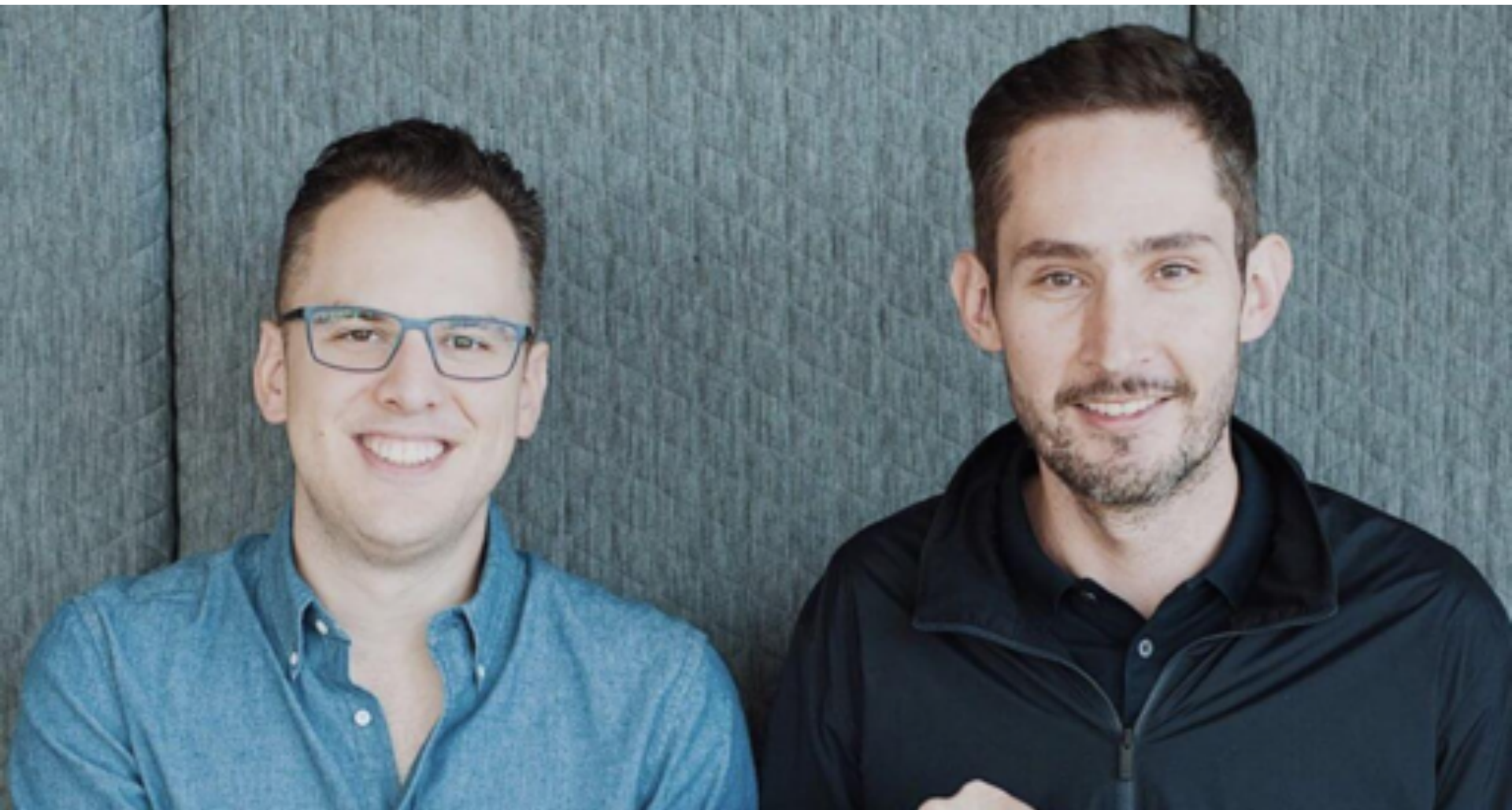
Sepia Example

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



Mike Krieger

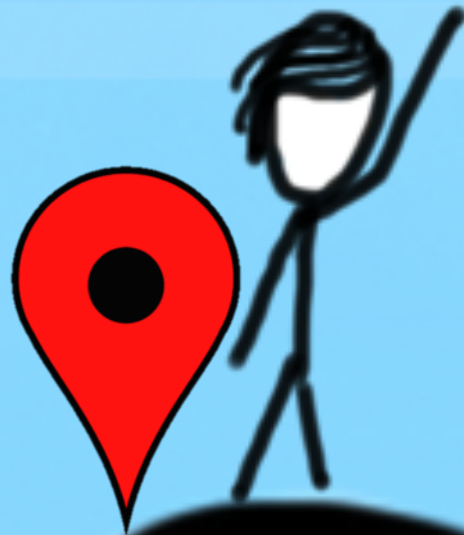
Kevin Systrom



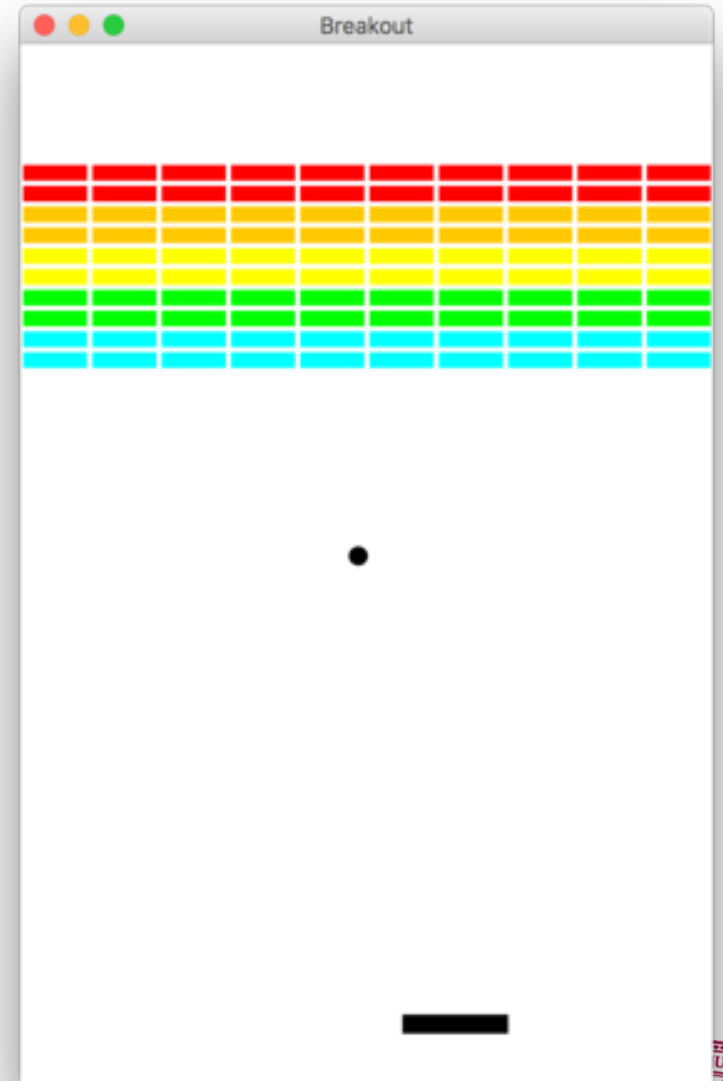
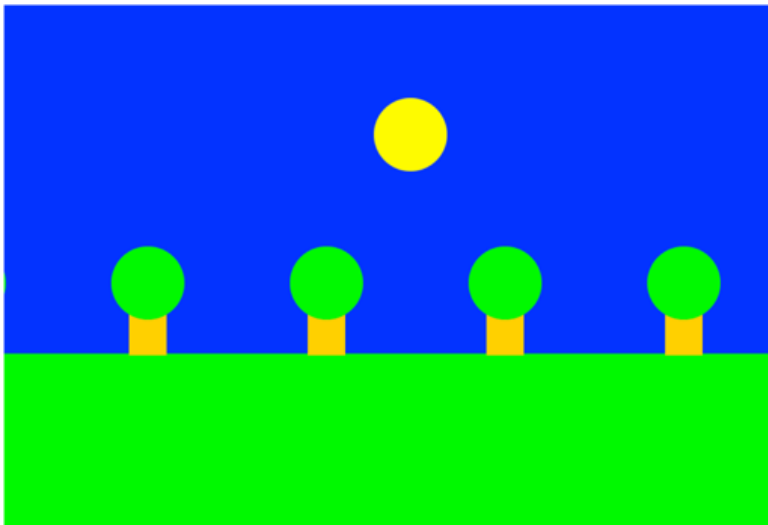
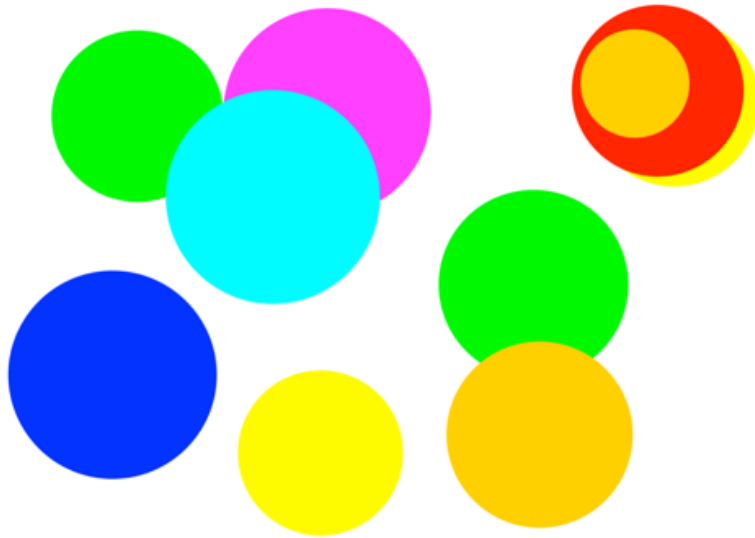
End Review

Today's Goal

1. How do I draw shapes?



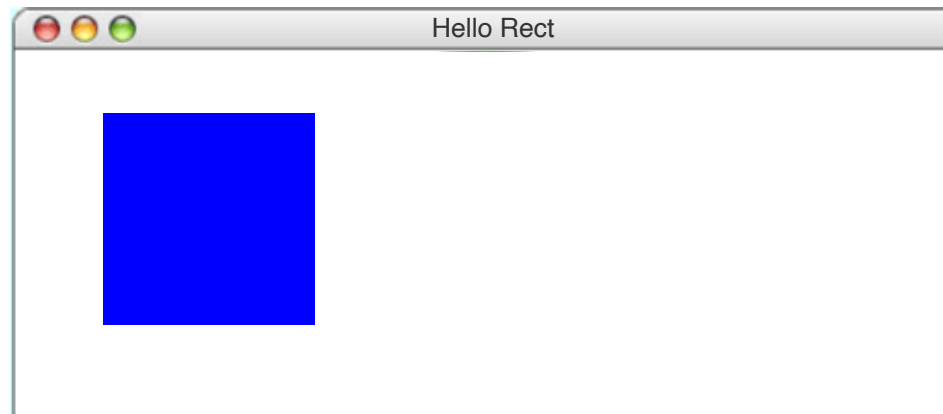
Graphics Programs



Draw a Rectangle

the following `main` method displays a blue square

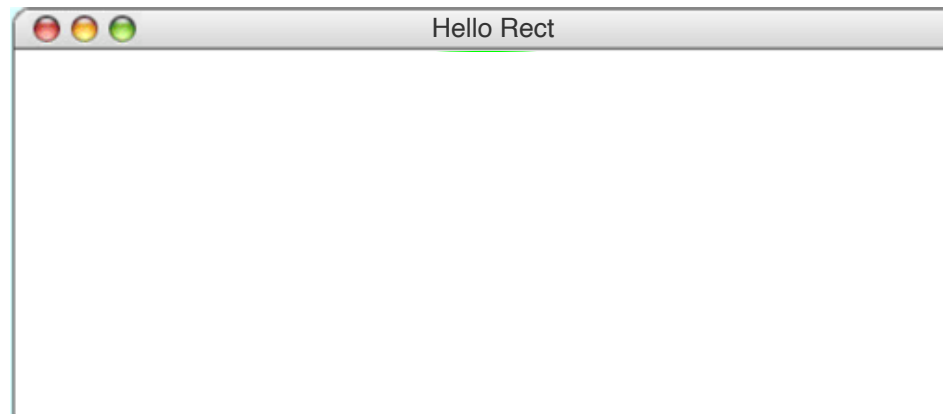
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')  
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")  
    canvas.mainloop()
```



Draw a Rectangle

the following `main` method displays a blue square

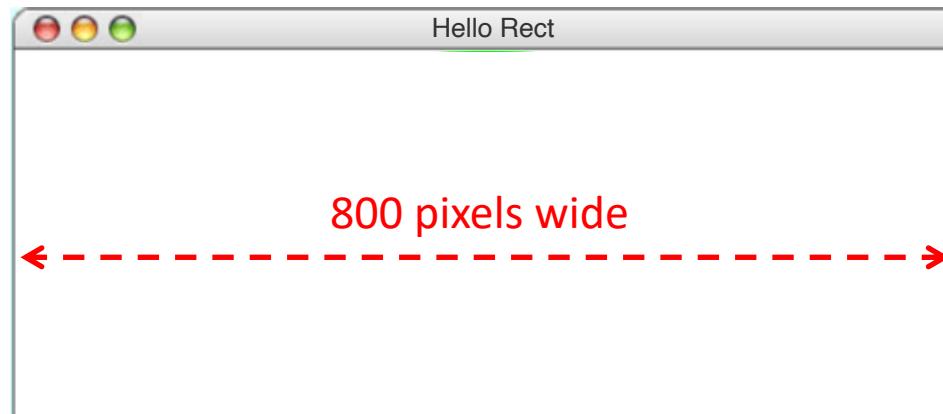
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

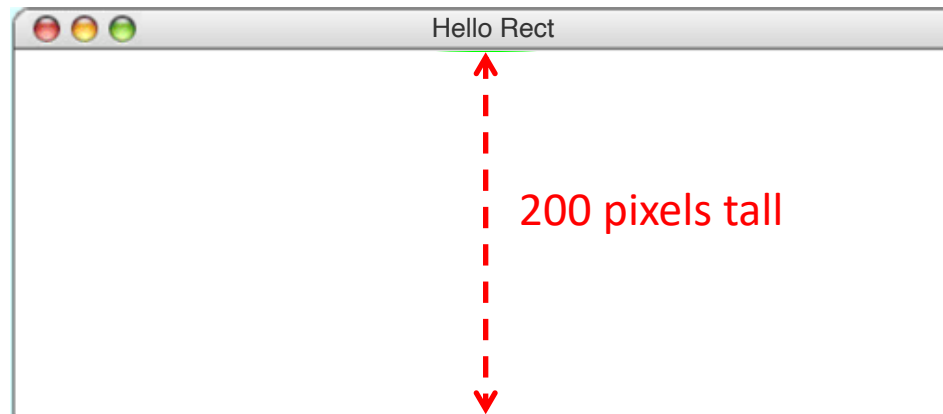
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')
```



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

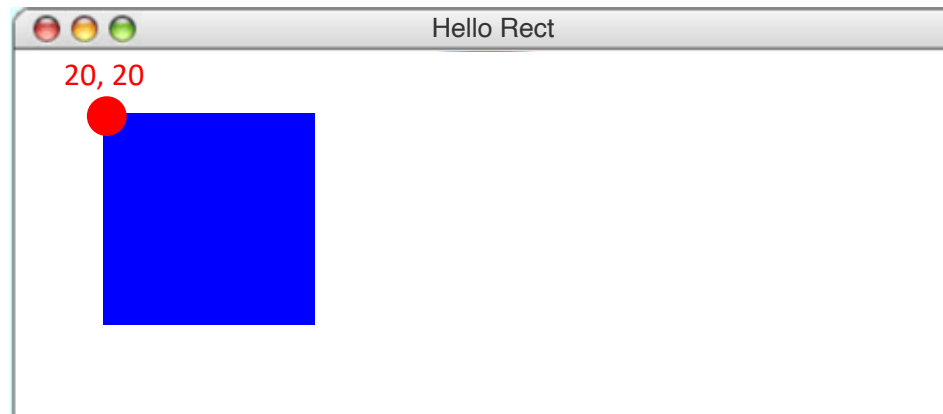
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

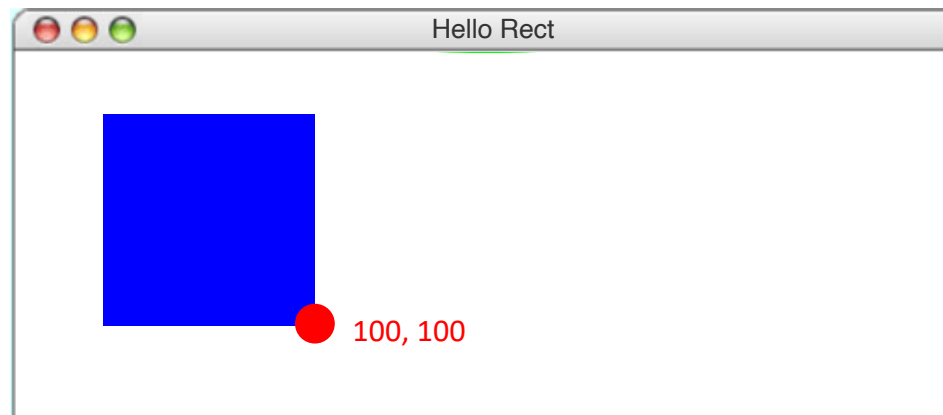
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')  
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

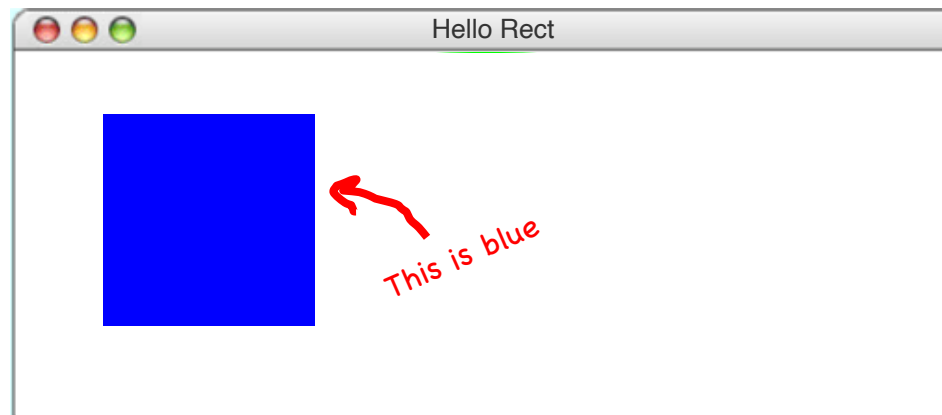
```
def main():  
    canvas = make_canvas(800, 200, 'Hello Rect')  
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



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")
```



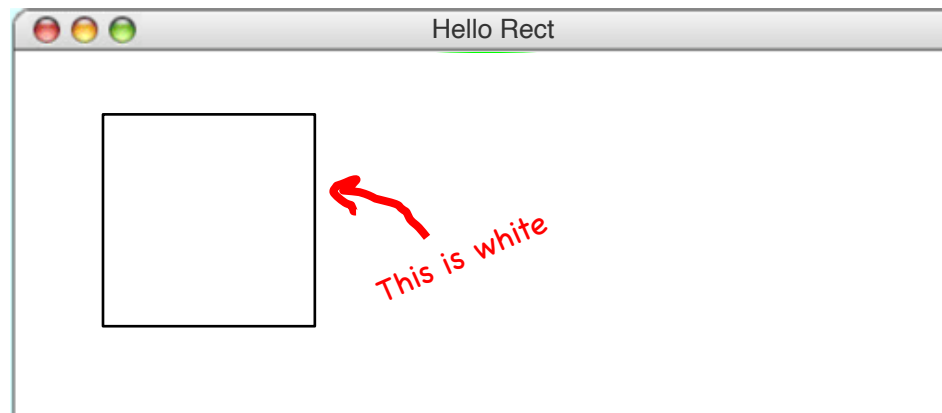
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



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)
```



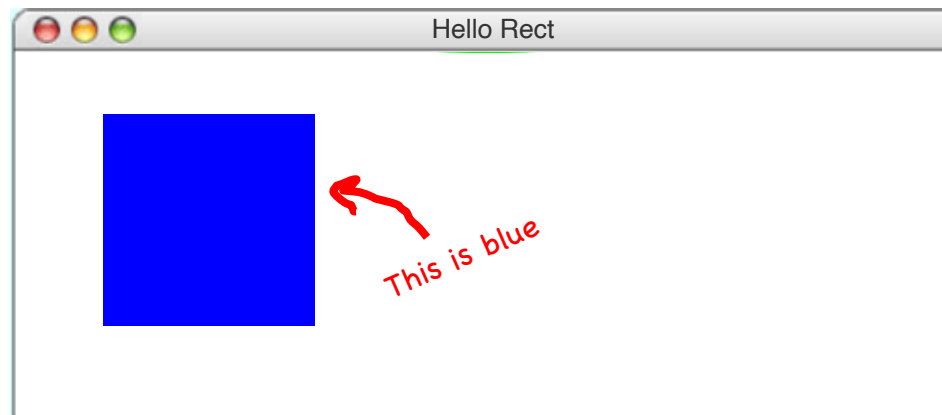
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



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")
```



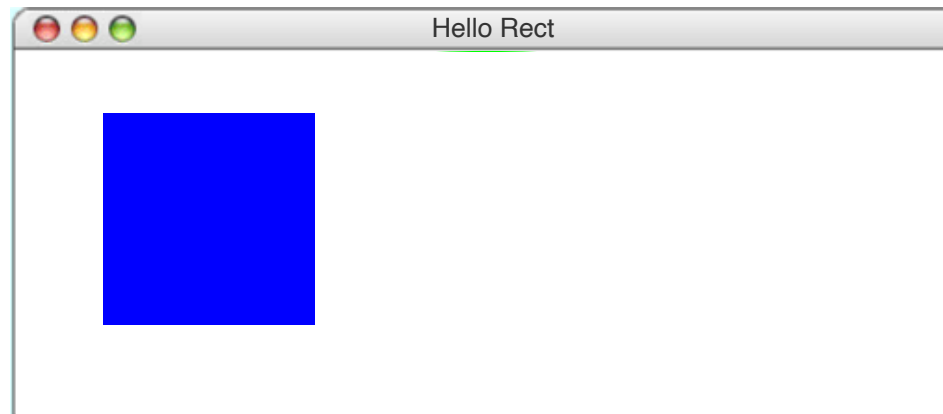
Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.



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

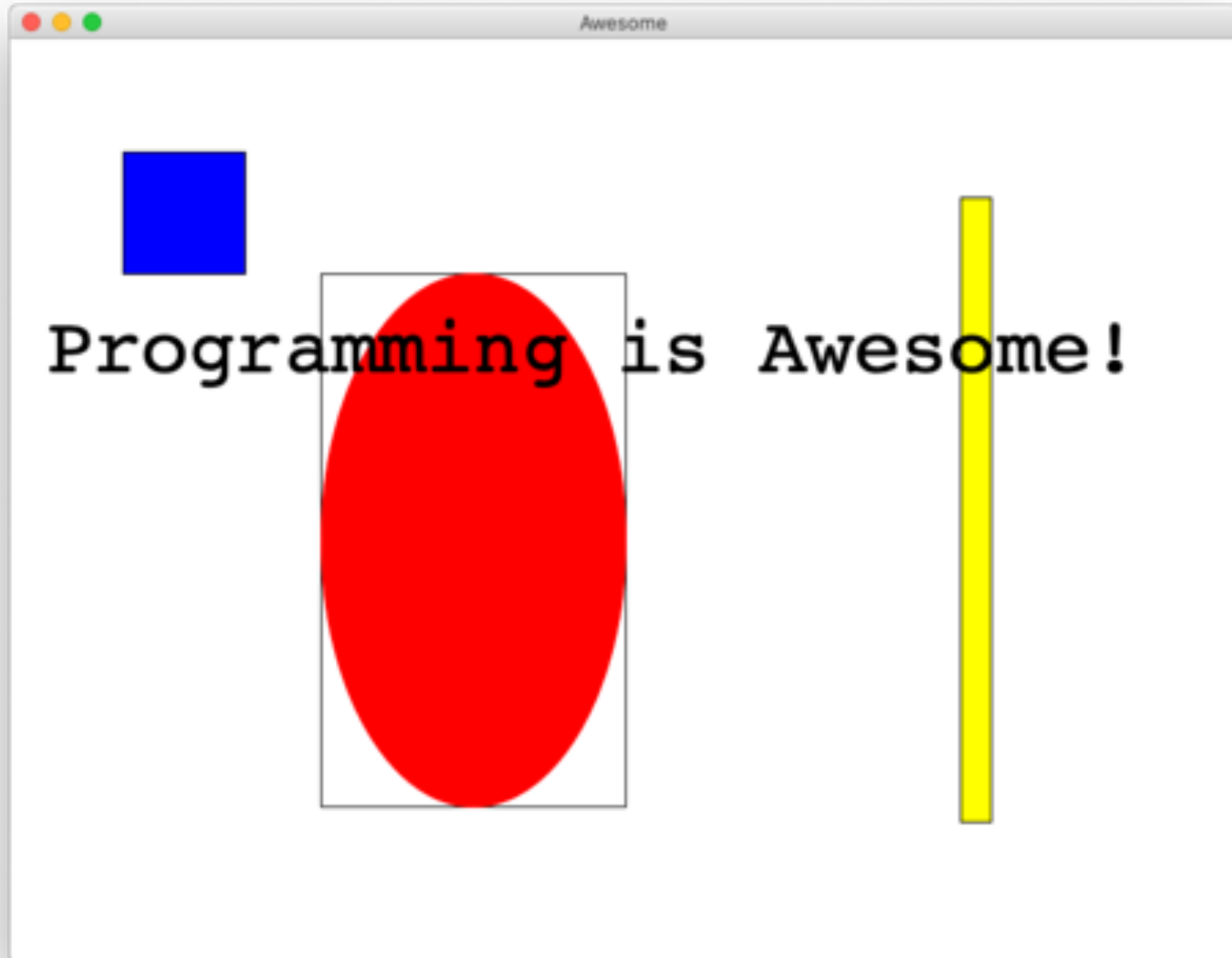
x 40,120

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()`

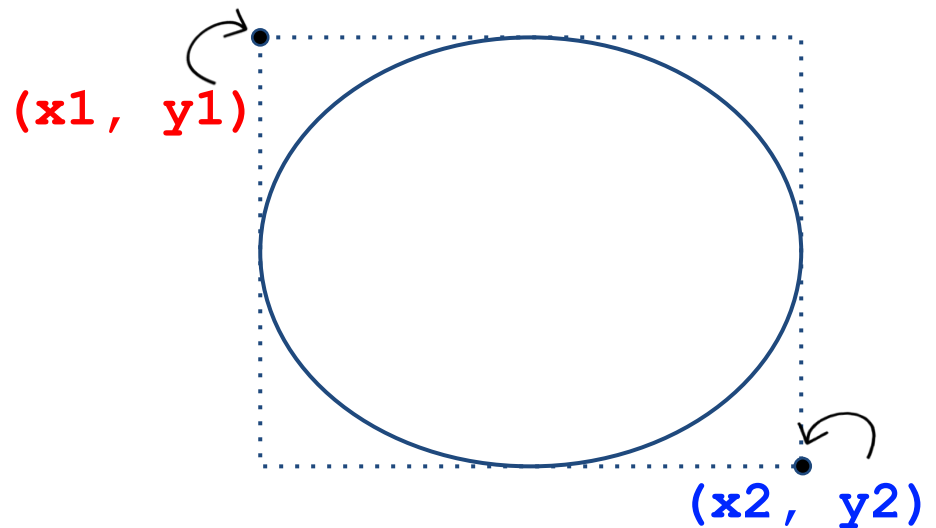
- `canvas.create_text()`



- `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()`

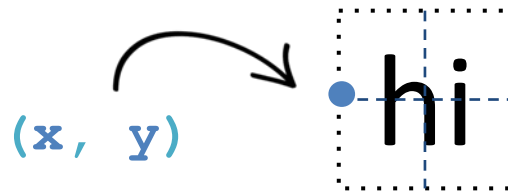


- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text(x, y, text='hi')`

hi



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



Pedagogy

CS106A

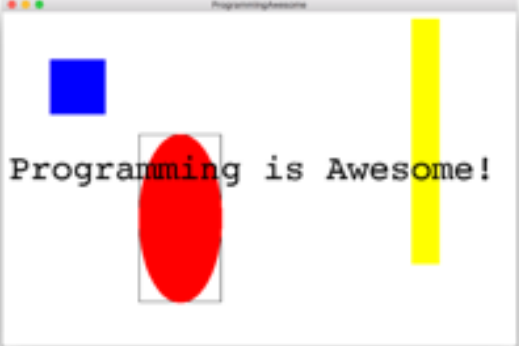
localhost:8000/examples/...

CS106A Lectures Handouts Assignments **Examples** Sections Schedule

Programming is Awesome

Written by Chris Piech

The point of this program is to show you a graphics program with each of the key shapes. We should draw two rectangles (one blue and one yellow), draw one red oval, with a black non-filled rectangle in the same location. In the center of the screen we write "Programming is Awesome." Here is what our program looks like:



```
Here is the corresponding code:
```

```
import acm.graphics.*;
import acm.program.*;
import java.awt.*;

public class ProgrammingIsAwesome extends GraphicsProgram {
    // draws the screen in the picture above
    public void run() {
        // half the height of the screen.
        double centerY = getHeight()/2;

        // make and add a blue square
        GRect blueSquare = new GRect(80, 80); // width and height are 80
        blueSquare.setFillColor(Color.BLUE); // make the square blue
        blueSquare.setFilled(true); // fill the square
        add(blueSquare, 70, 70); // add the square to the screen

        // add a long yellow rectangle
        GRect yellowRect = new GRect(80, 180);
        yellowRect.setFillColor(Color.YELLOW);
        yellowRect.setFilled(true);
        add(yellowRect, 400, 100);

        // make and add a red oval
        GOval redOval = new GOval(120, centerY); // width and height
        redOval.setFillColor(Color.RED);
        redOval.setFilled(true);
        add(redOval, 200, 180); // add to location (200, 180)

        // make and add a rectangle which fits around the red oval
        GRect circOutline = new GRect(120, centerY);
        add(circOutline, 200, 180);
    }
}
```

CS106A

localhost:8000

CS106A Lectures Handouts Assignments Examples Sections

CS106A: Programming Methodologies

Stanford University: Winter 2019
Monday, Wednesday, Friday 3:30pm - 4:20pm in NVIDIA Auditorium

RESOURCES

- CS106A Info
- Course Schedule
- Style Guide
- Piazza
- Eclipse
- Stanford Java Lib**
- Blank Karel Project
- Videos
- LalR Hours

ASSIGNMENTS

- Assn 2: Simple Java
- Assn 1: Karel

EXAMS

ANNOUNCEMENTS

Section Assignments & Late Signups

3 days ago

For those who submitted section preferences by 5PM on Sunday, we making section assignments; as a reminder, sections start this week! your assigned section via cs198.stanford.edu). If you were unable to su by the 5PM Sunday deadline, the late signup form is available on th once you log in.

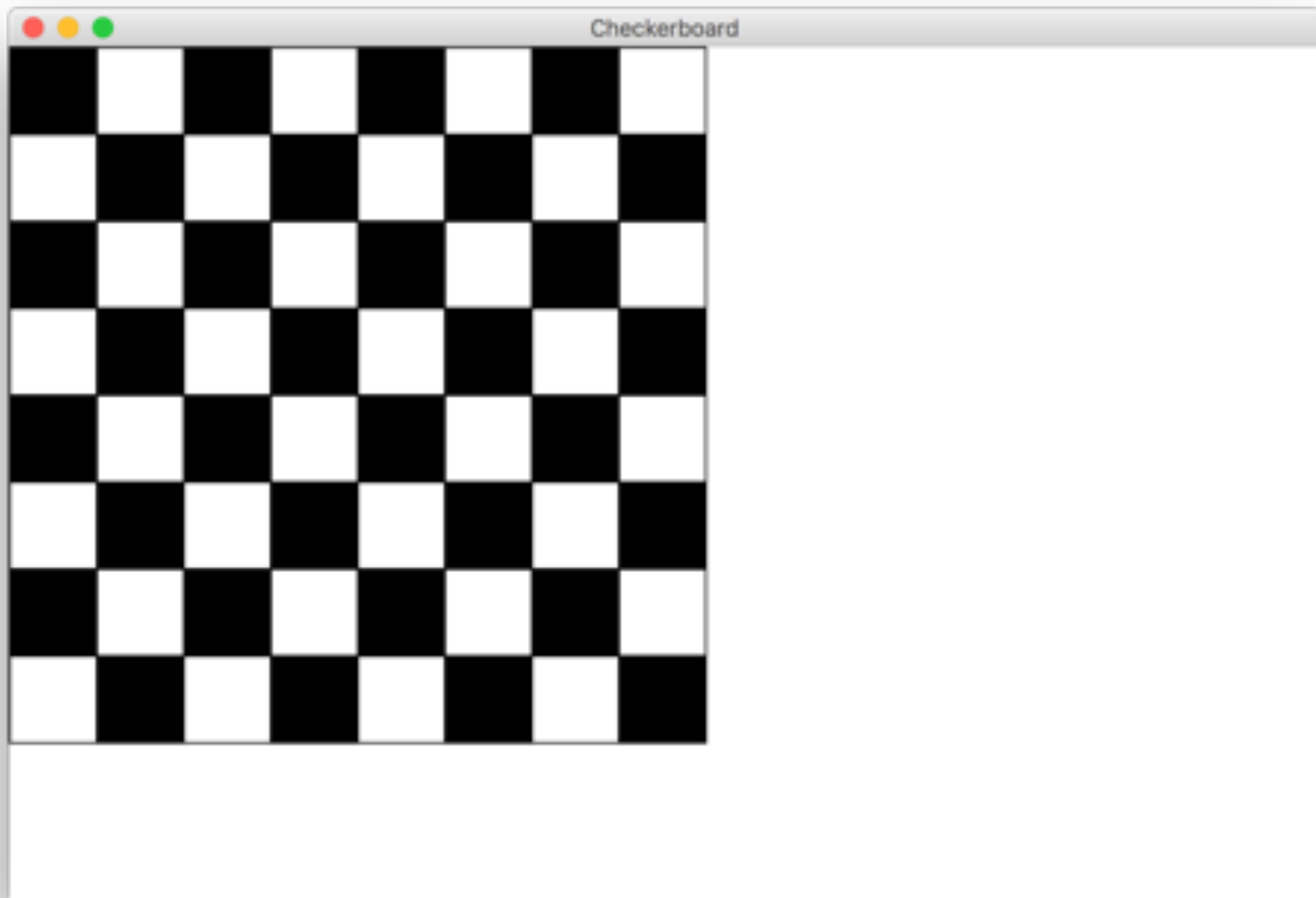
If you would like to **individually** switch to a different section because or other constraints please request a swap via cs198.stanford.edu. Yo this form to join the section of a partner who is another section. If questions, email Brahm.

Section Signups Open Until 5PM Sunday 1/13

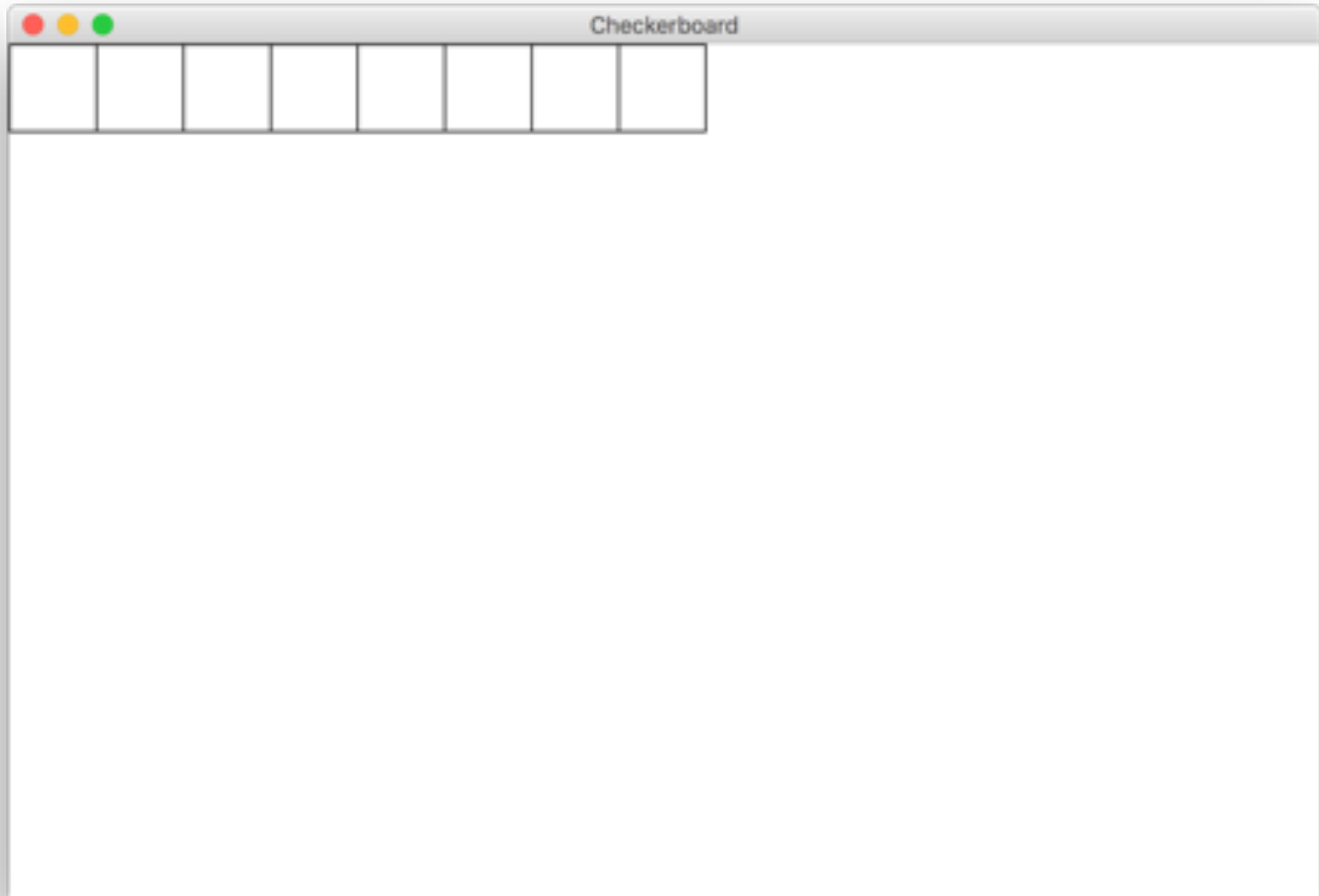
8 days ago

Section signups are now open! Click on the "Section" tab at the t "Section Signup" to submit your preferences. As a reminder, signup **come first serve**. As such, you may modify your preferences any tim Sunday 5PM deadline. We will notify you of your section assignment ea

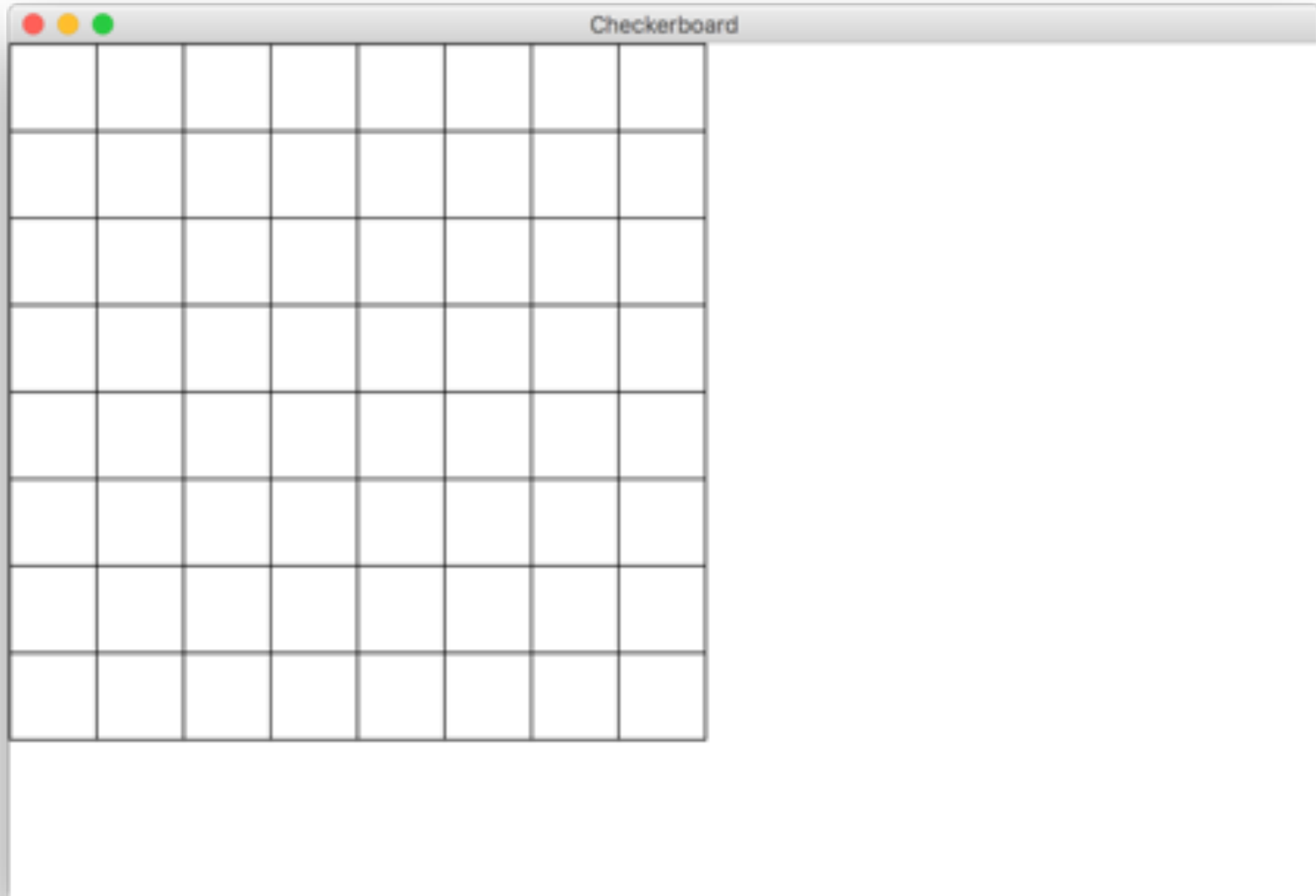
Goal



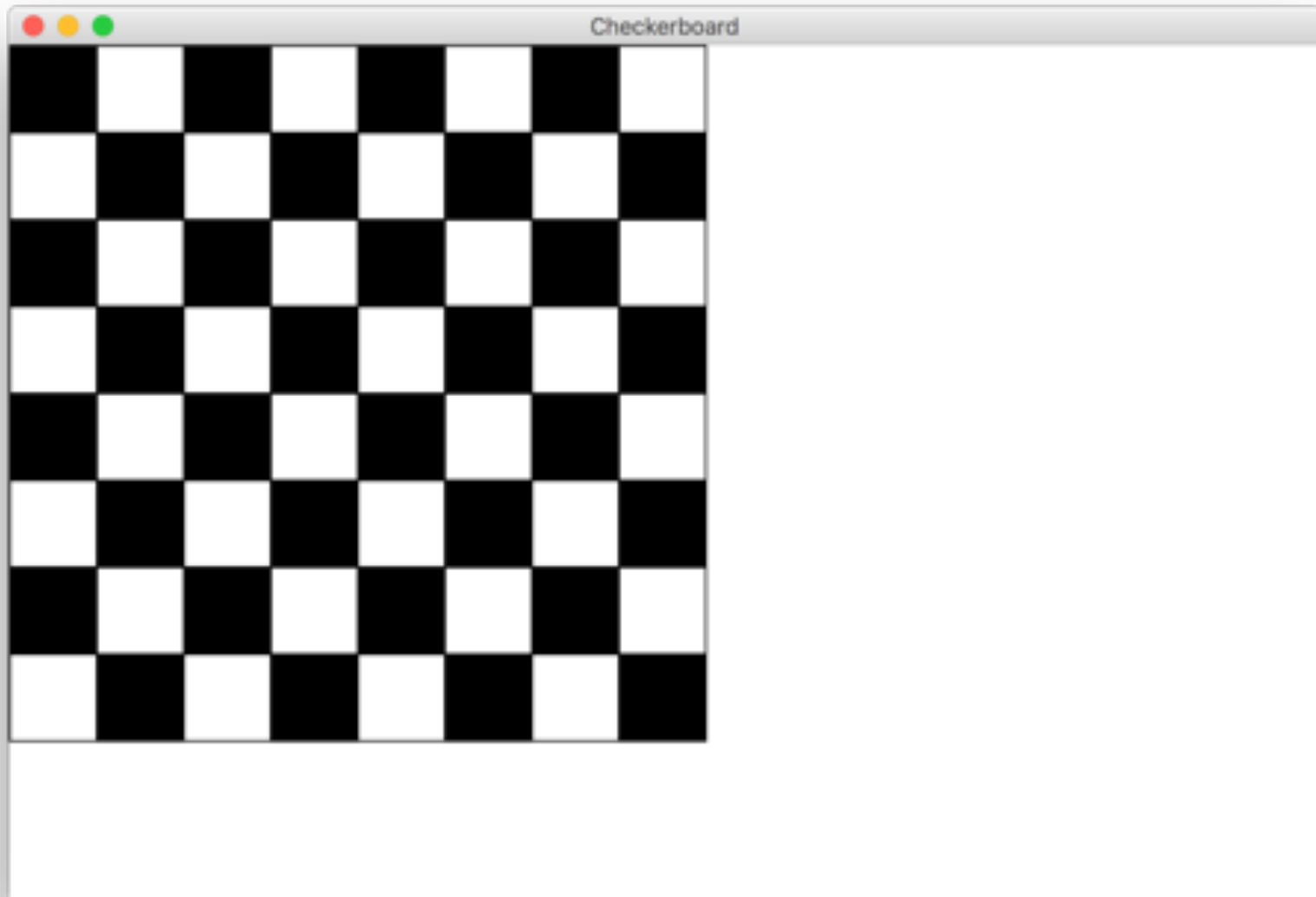
Milestone 1

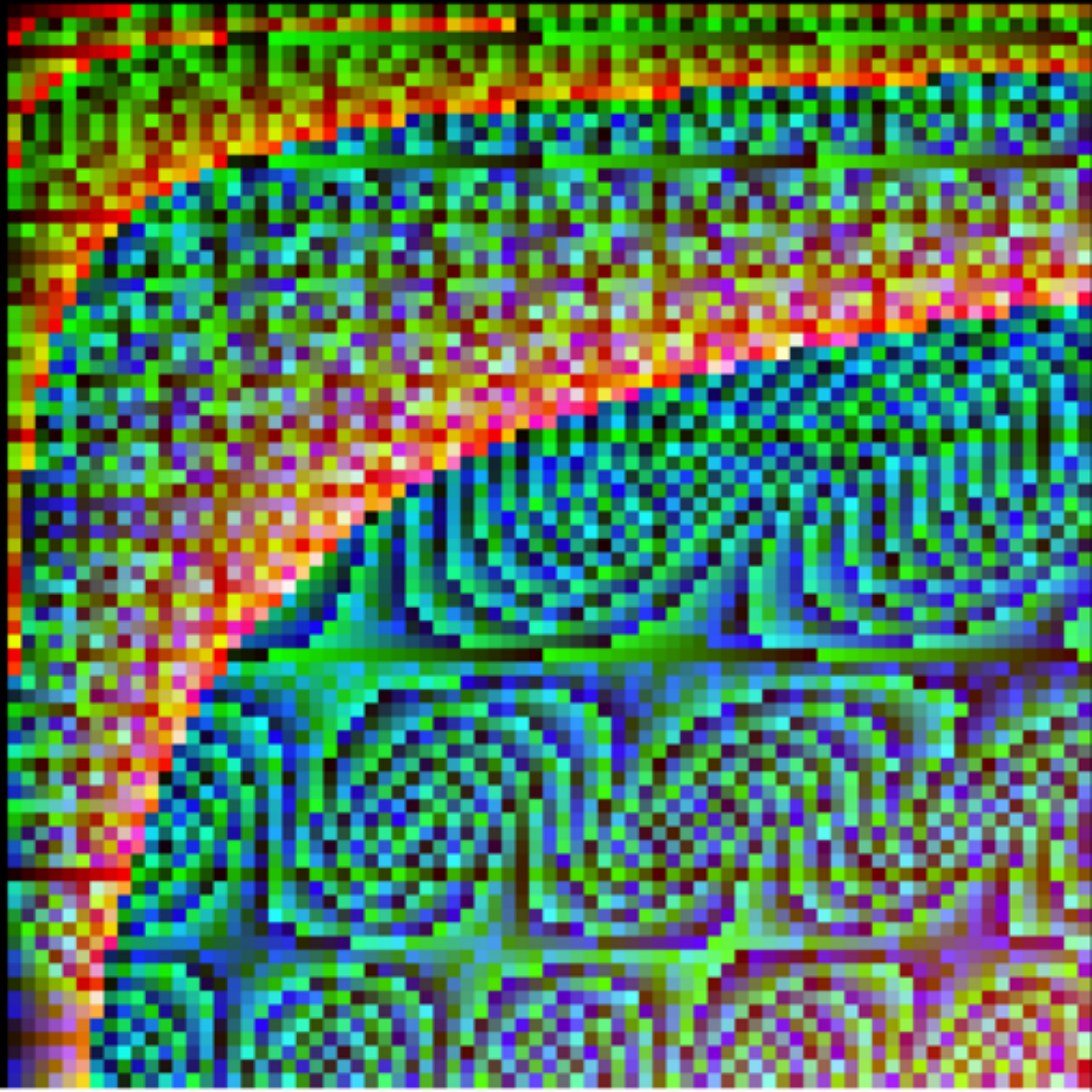


Milestone 2



Milestone 3





Teaser for next week...

Hold up!

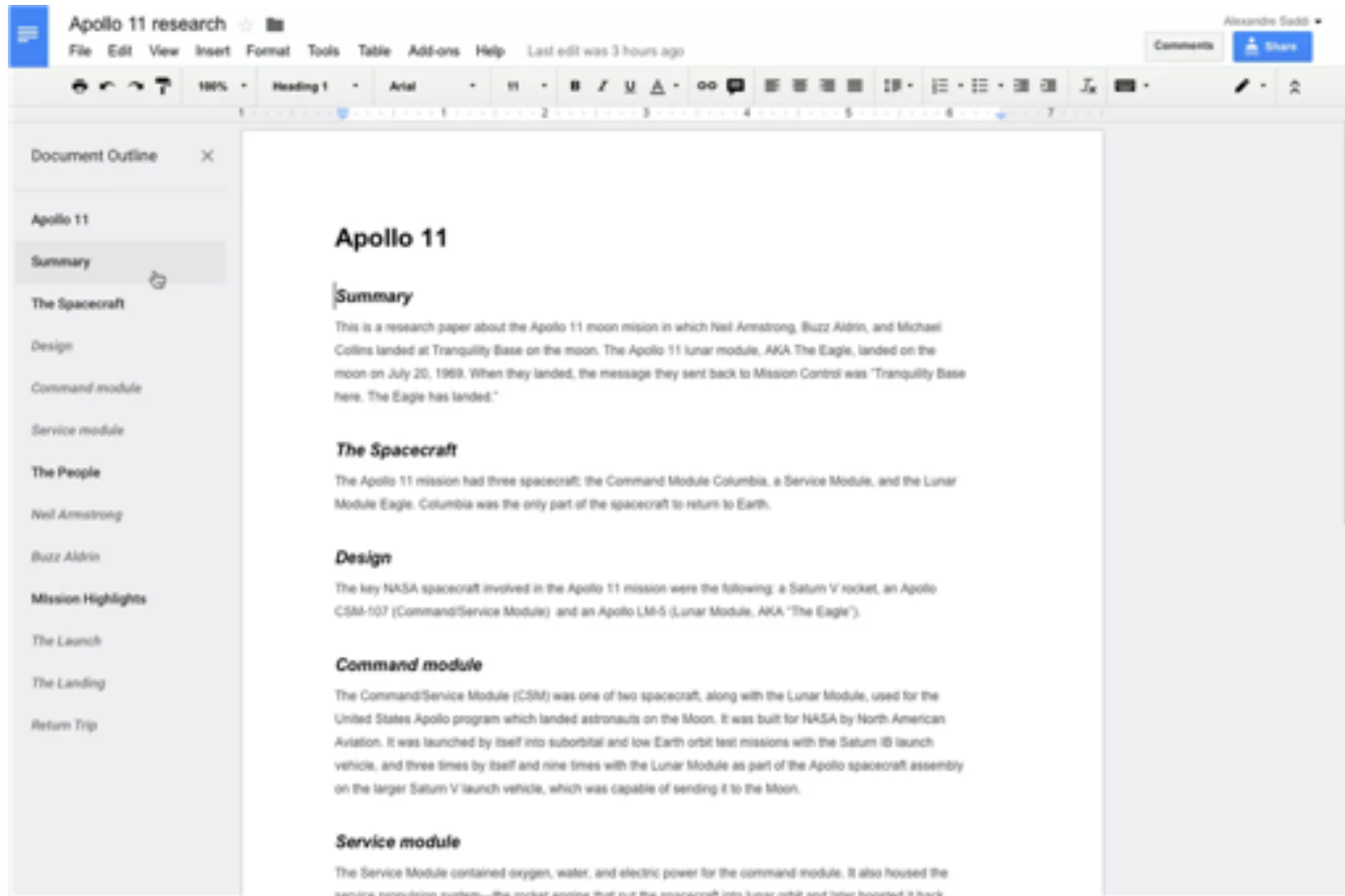
```
def draw_square(canvas, row, col):
```

*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/1eBtnEill3KHe_fFS-kSAOpXqeSXpbfTTMImOgj6I9dvk/



```
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(...)
```

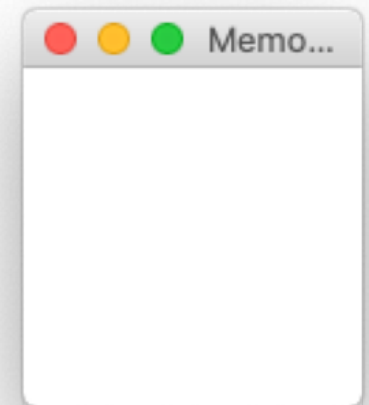
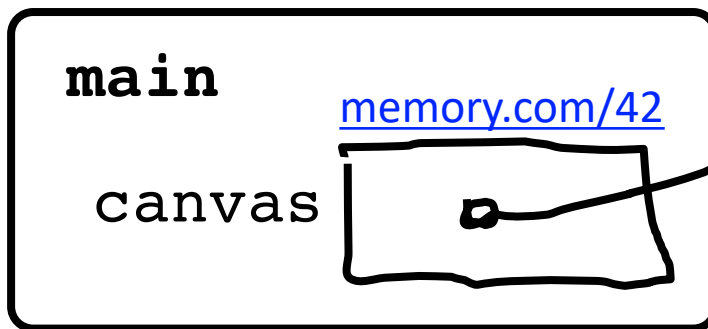
```
    draw_square(canvas)
```

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

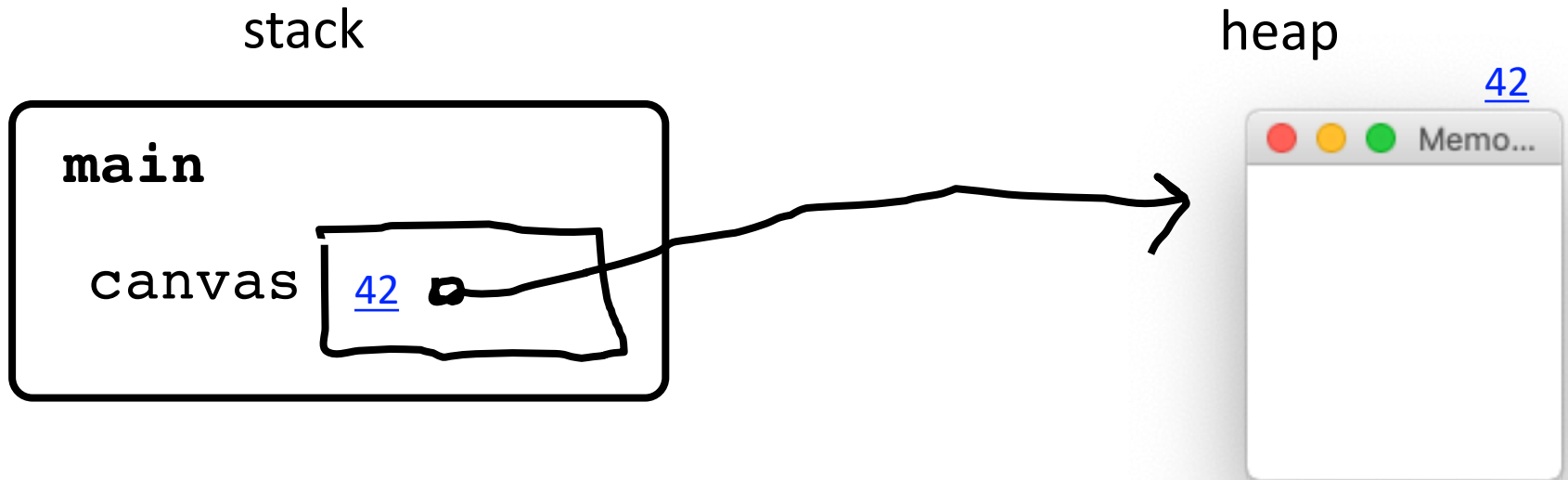
```
    canvas.create_rectangle(20, 20, 100, 100)
```

stack

heap

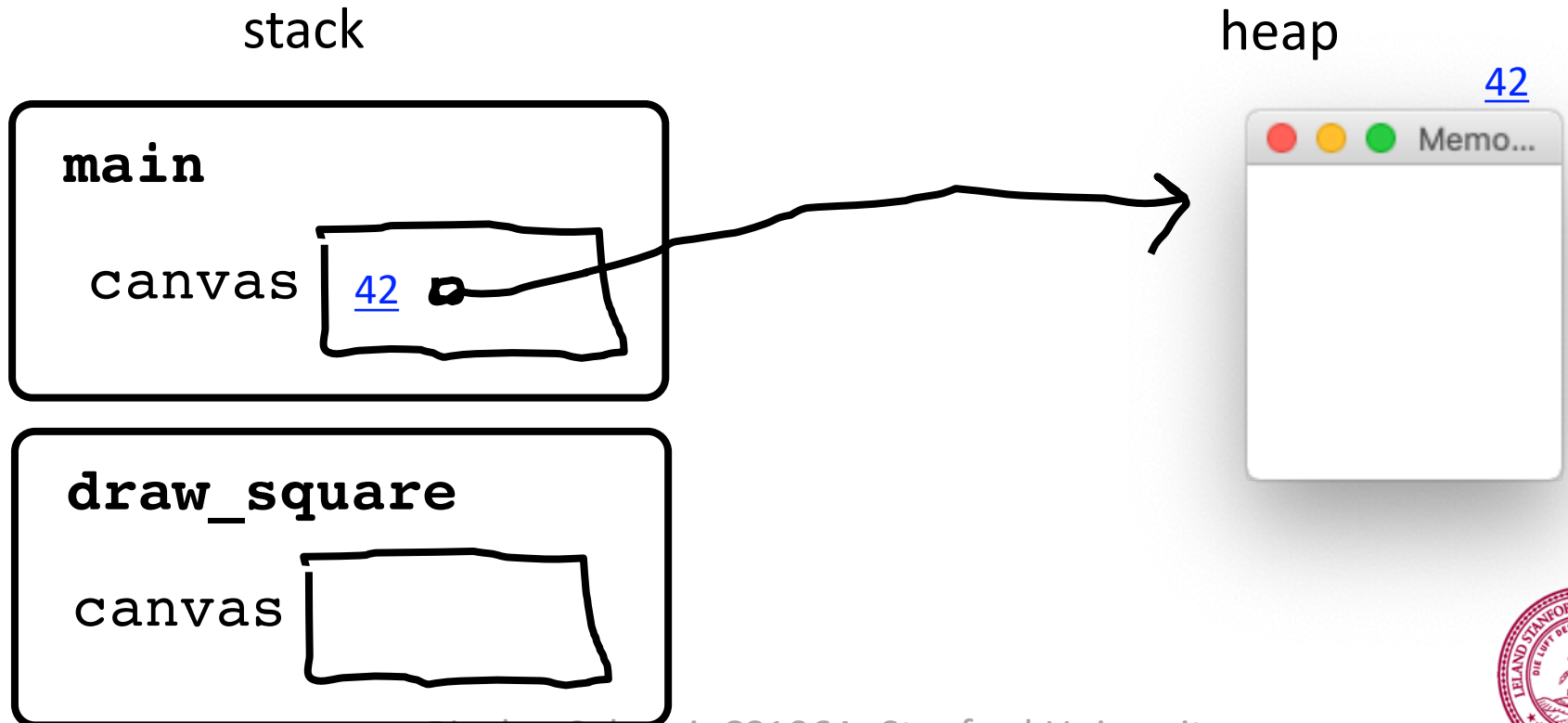


```
def main():  
    canvas = make_canvas(...)  
    draw_square(canvas)  
  
def draw_square(canvas):  
    canvas.create_rectangle(20, 20, 100, 100)
```

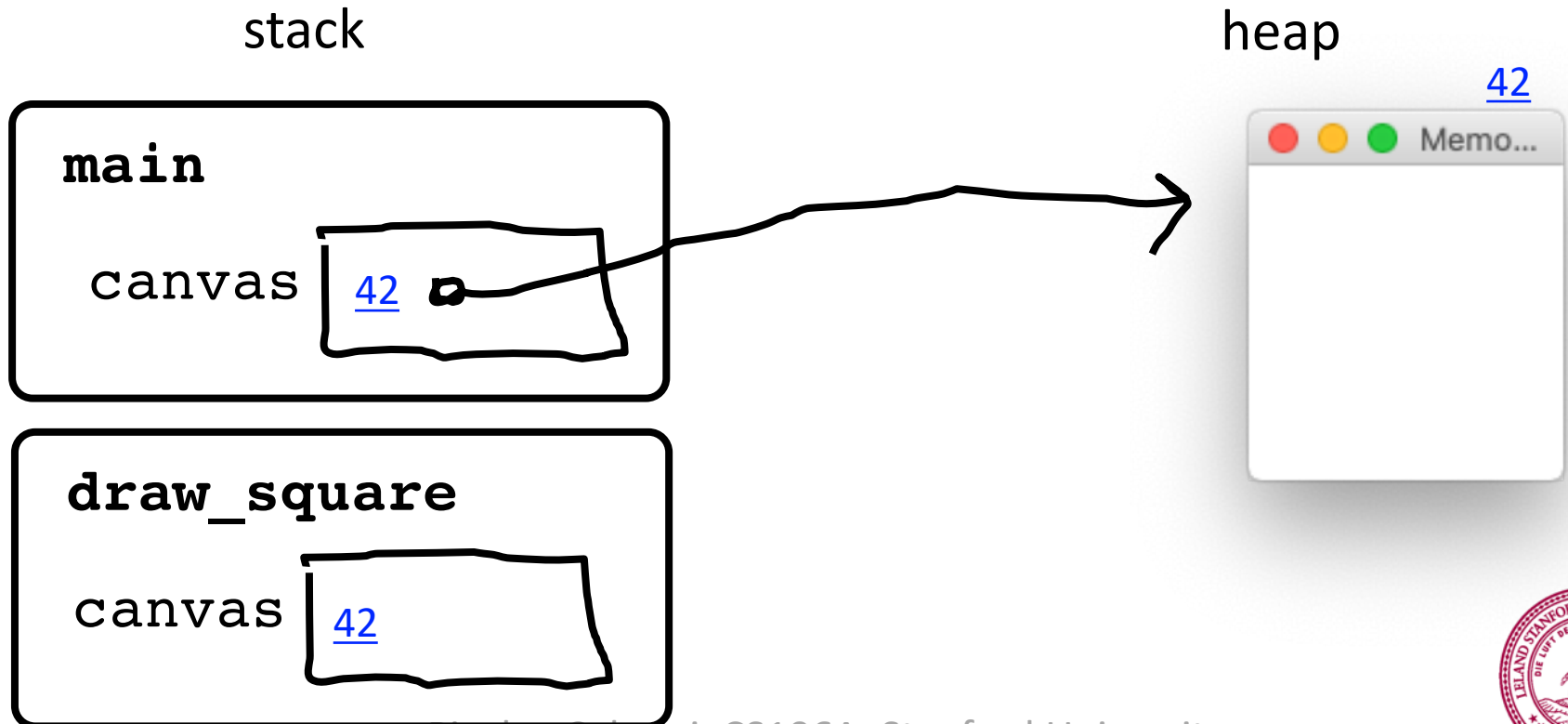


```
def main():  
    canvas = make_canvas(...)  
    draw_square(canvas)
```

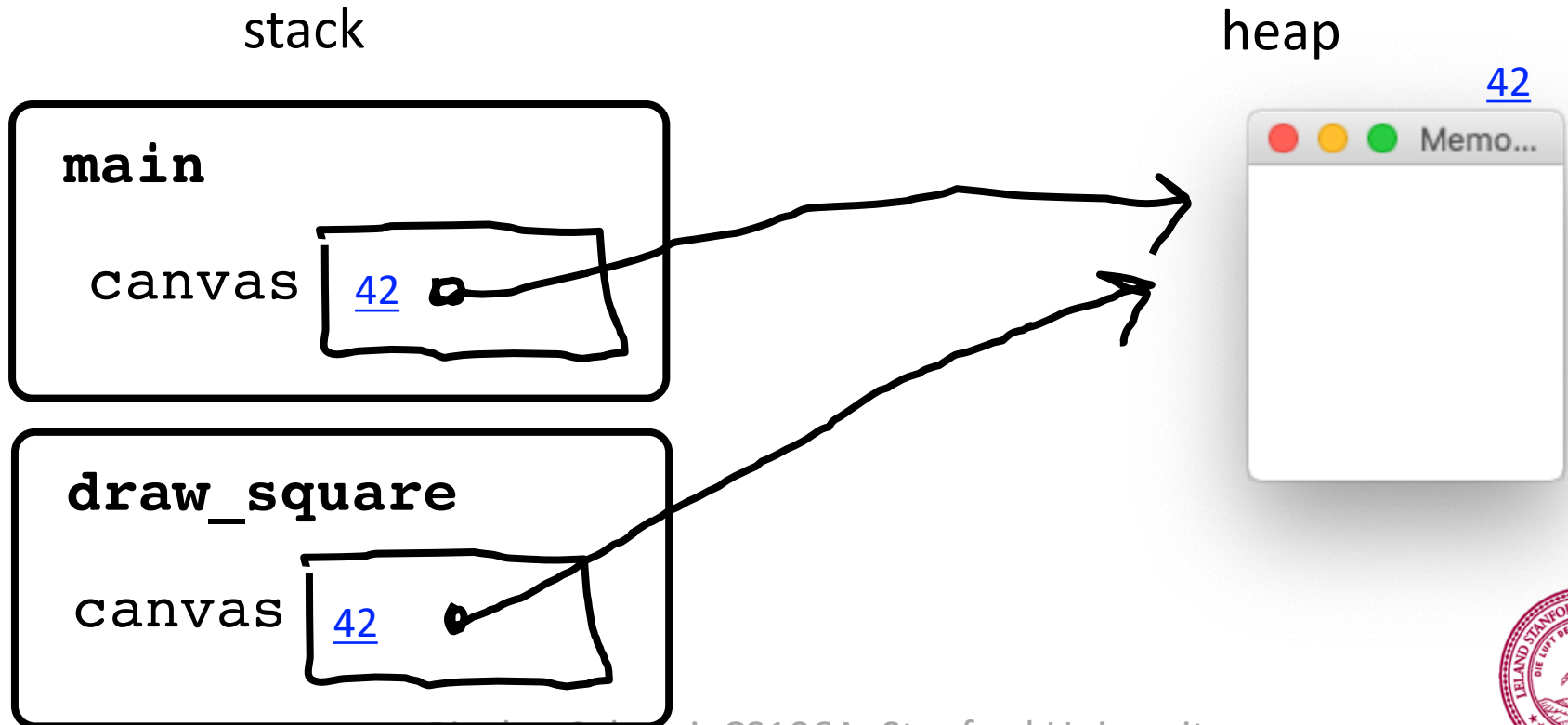
```
def draw_square(canvas):  
    canvas.create_rectangle(20, 20, 100, 100)
```



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

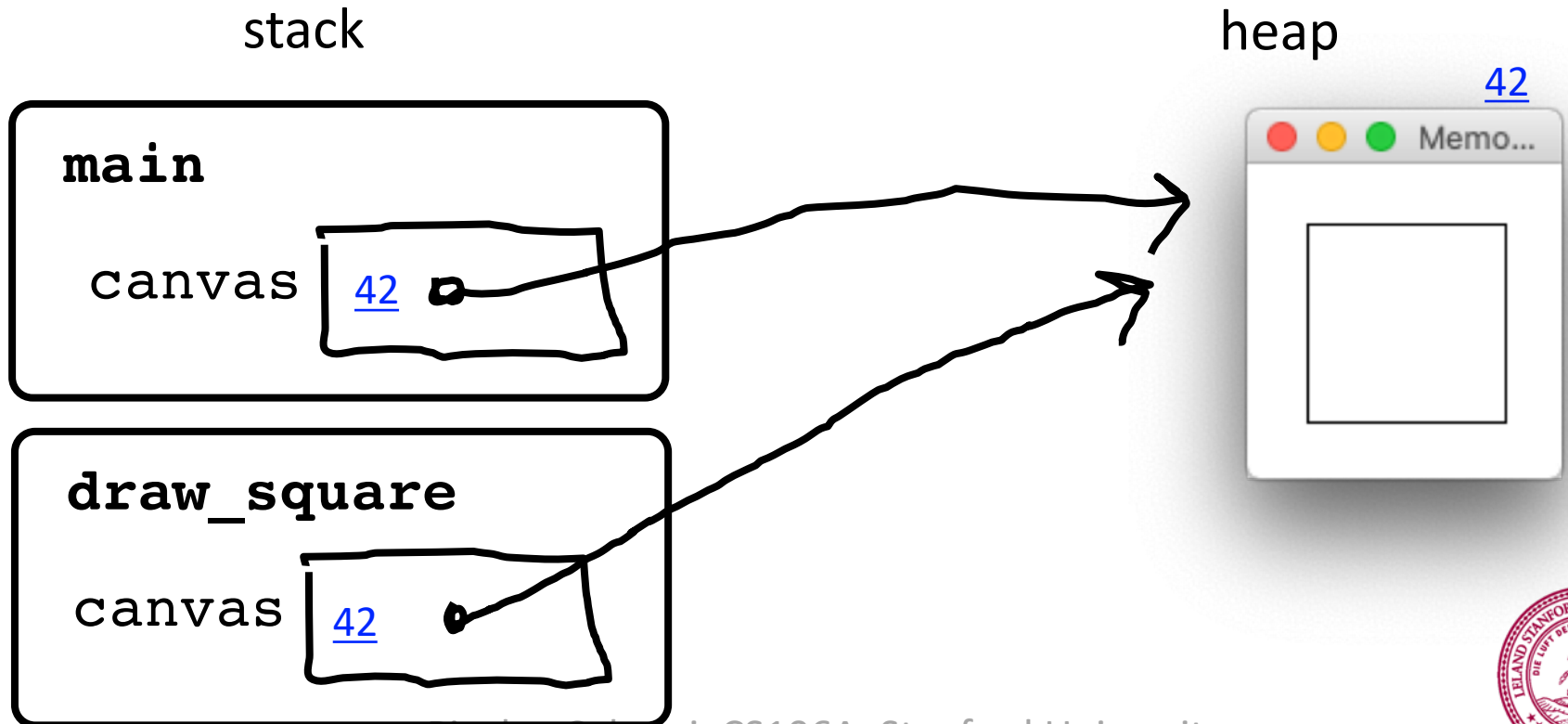


```
def main():  
    canvas = make_canvas(...)  
    draw_square(canvas)  
  
def draw_square(canvas):  
    canvas.create_rectangle(20, 20, 100, 100)
```



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

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

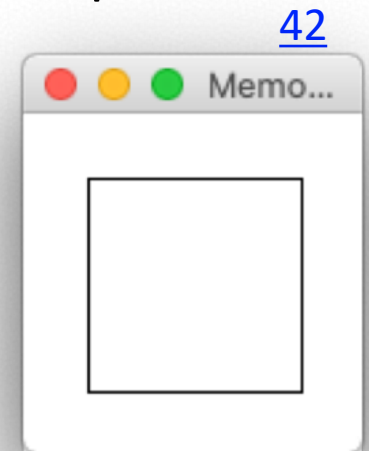
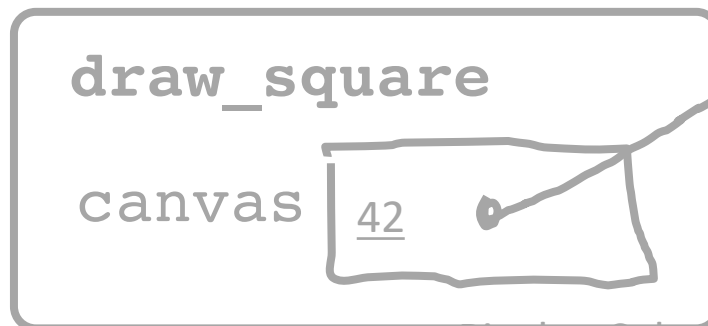
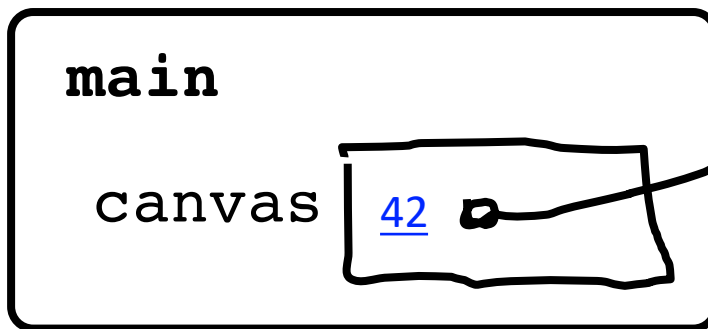


```
def main():  
    canvas = make_canvas(...)  
    draw_square(canvas)  
  
def draw_square(canvas):  
    canvas.create_rectangle(20, 20, 100, 100)
```

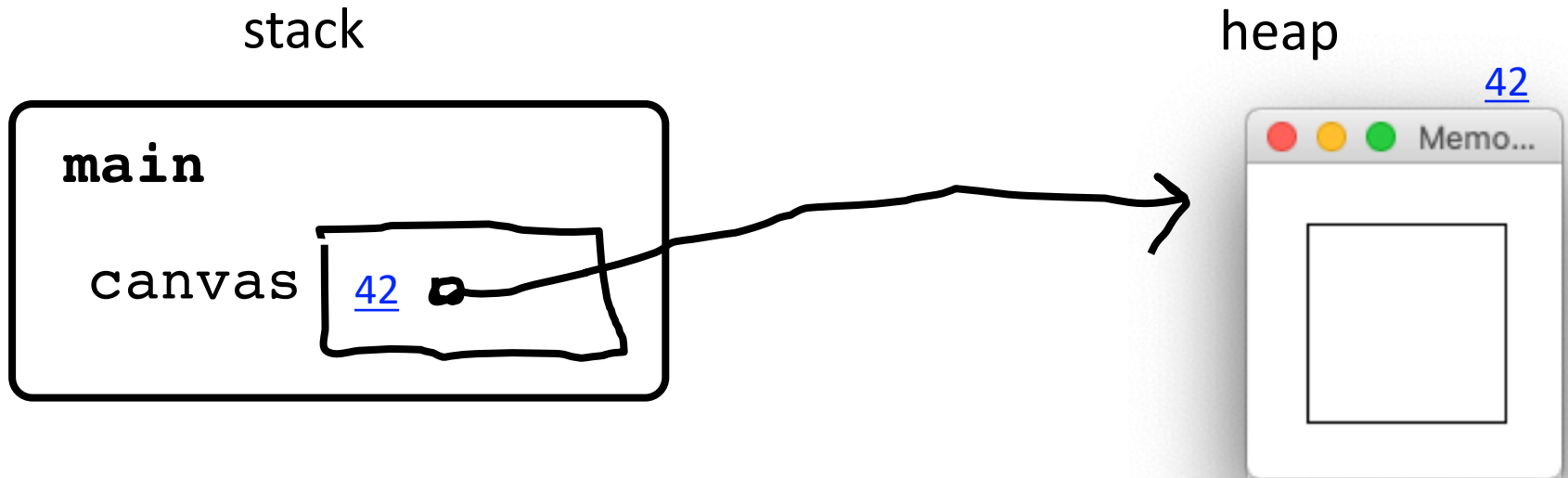


stack

heap



```
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)

