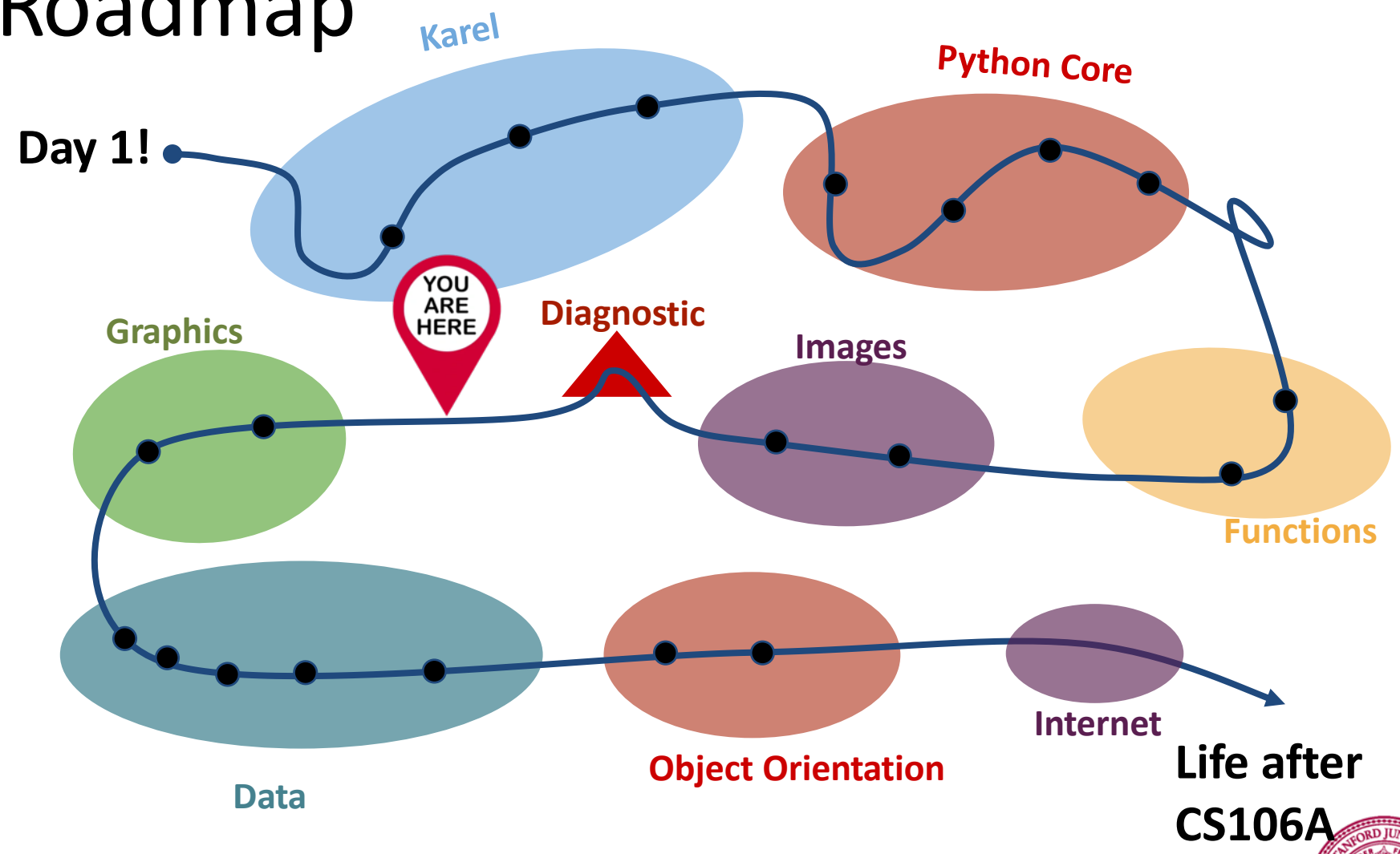
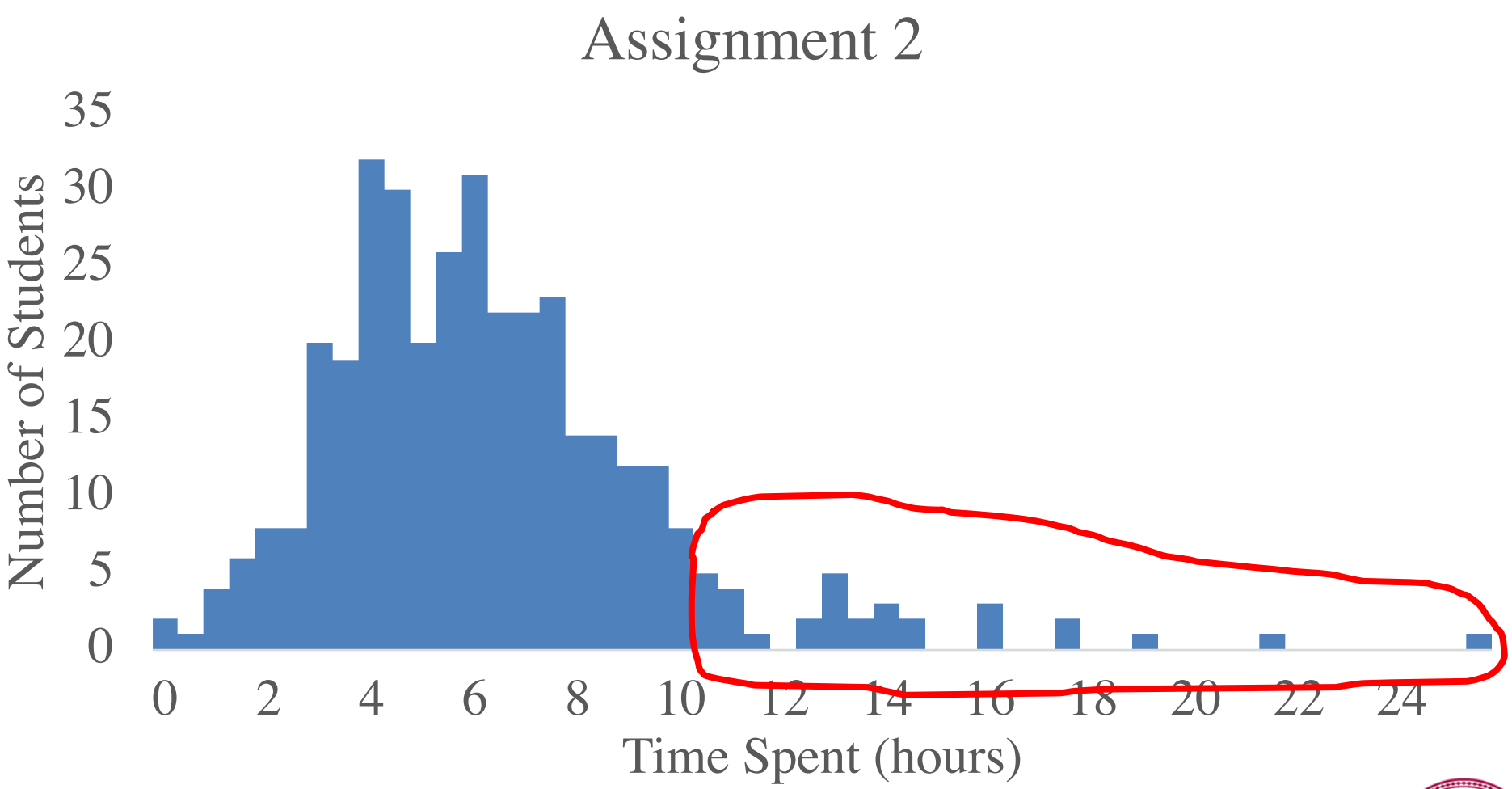
Graphics

Chris Piech and Mehran Sahami
CS106A, Stanford University

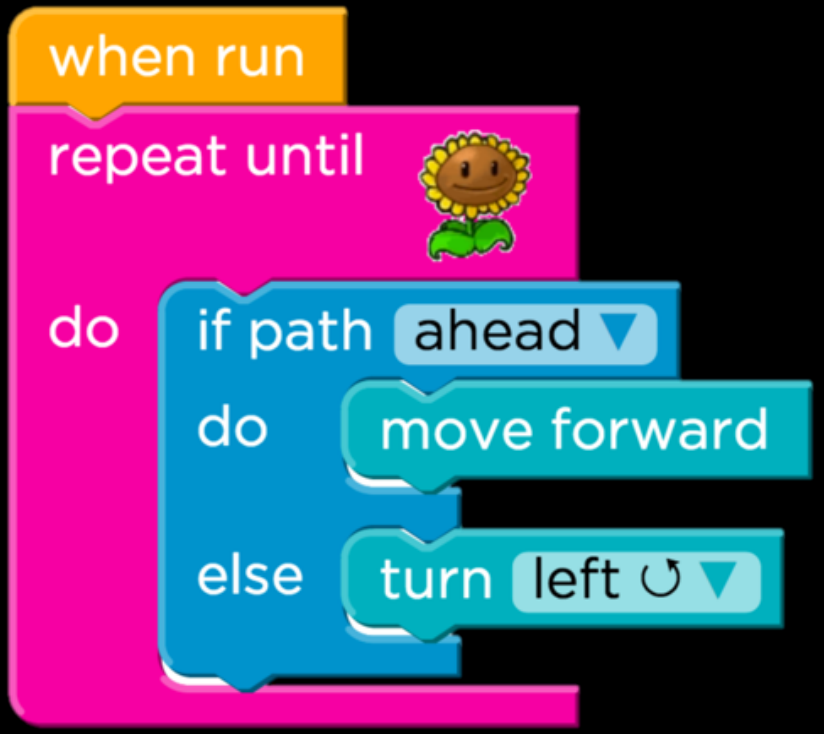
Roadmap


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

Learn by Doing



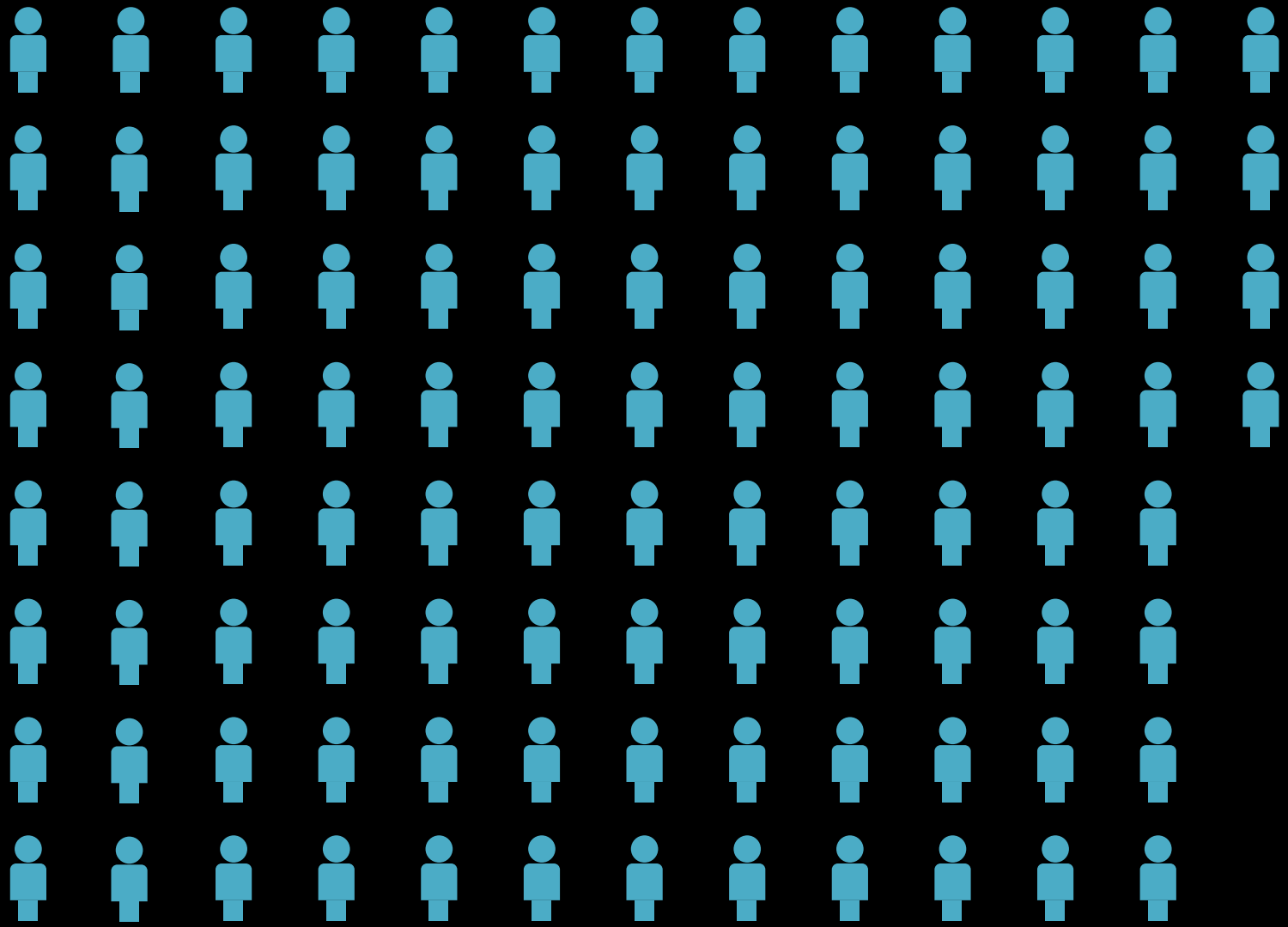
Learning to Program on the Internet



Task

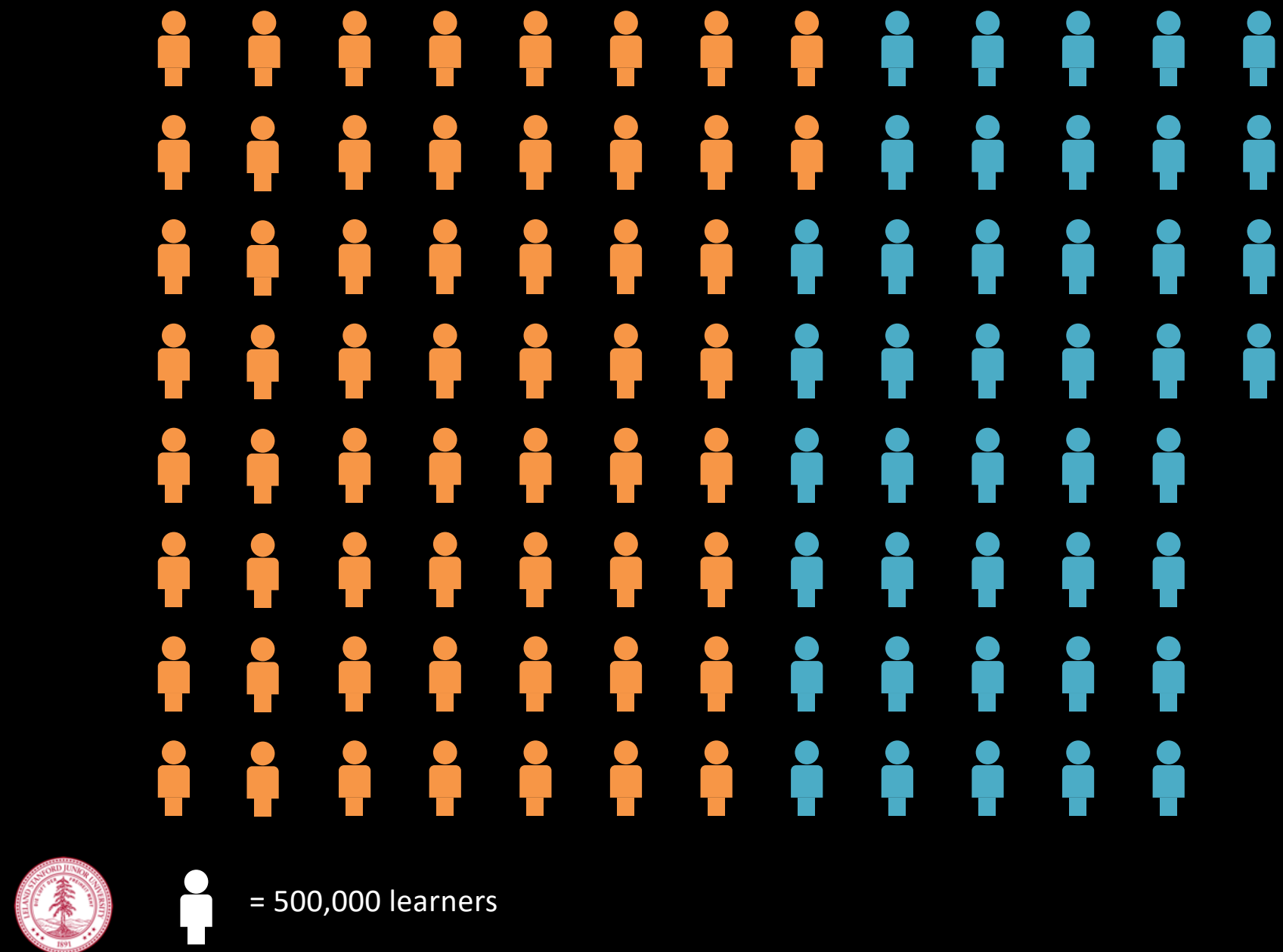
Almost a hundred thousand unique solutions

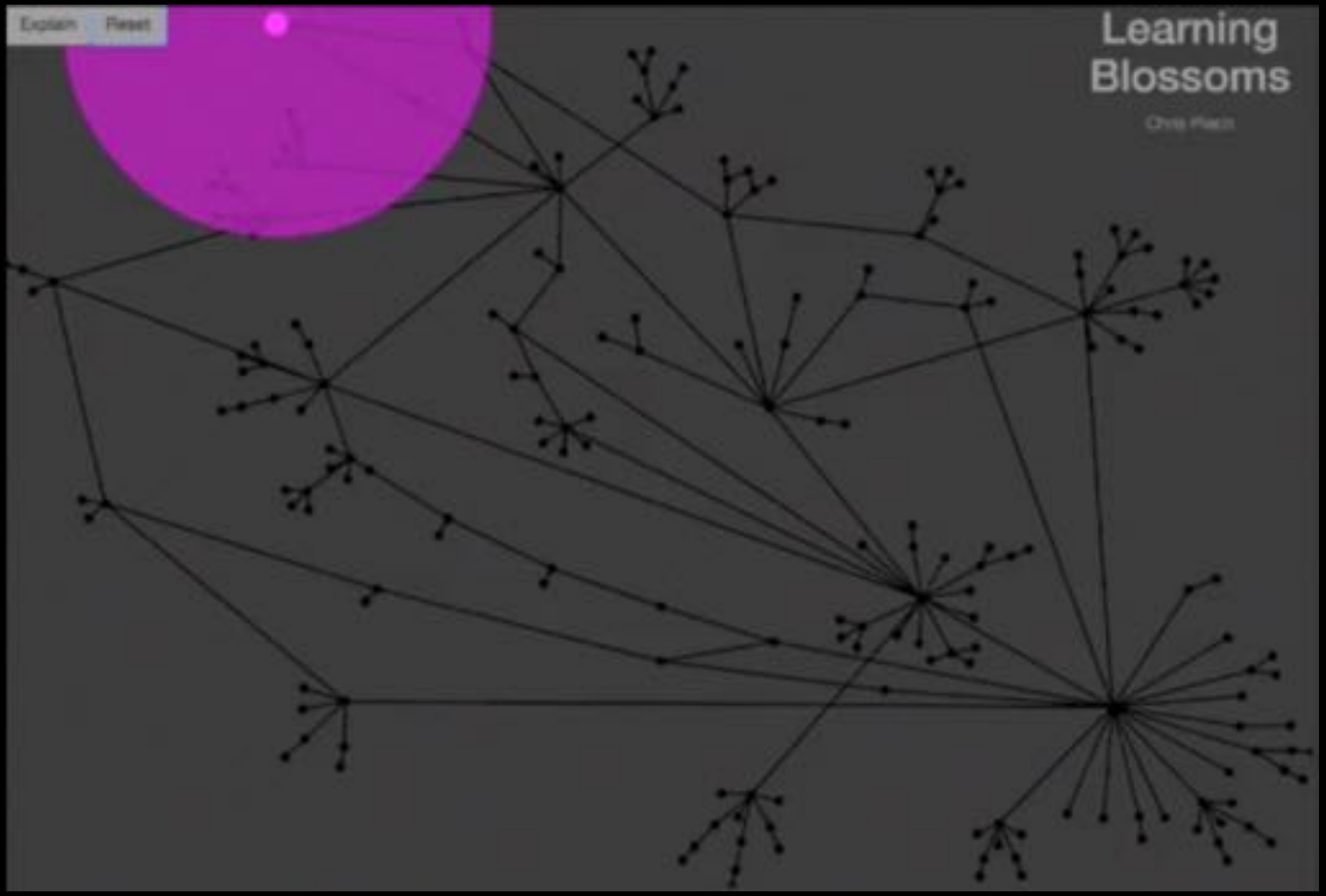
US K-12 Students



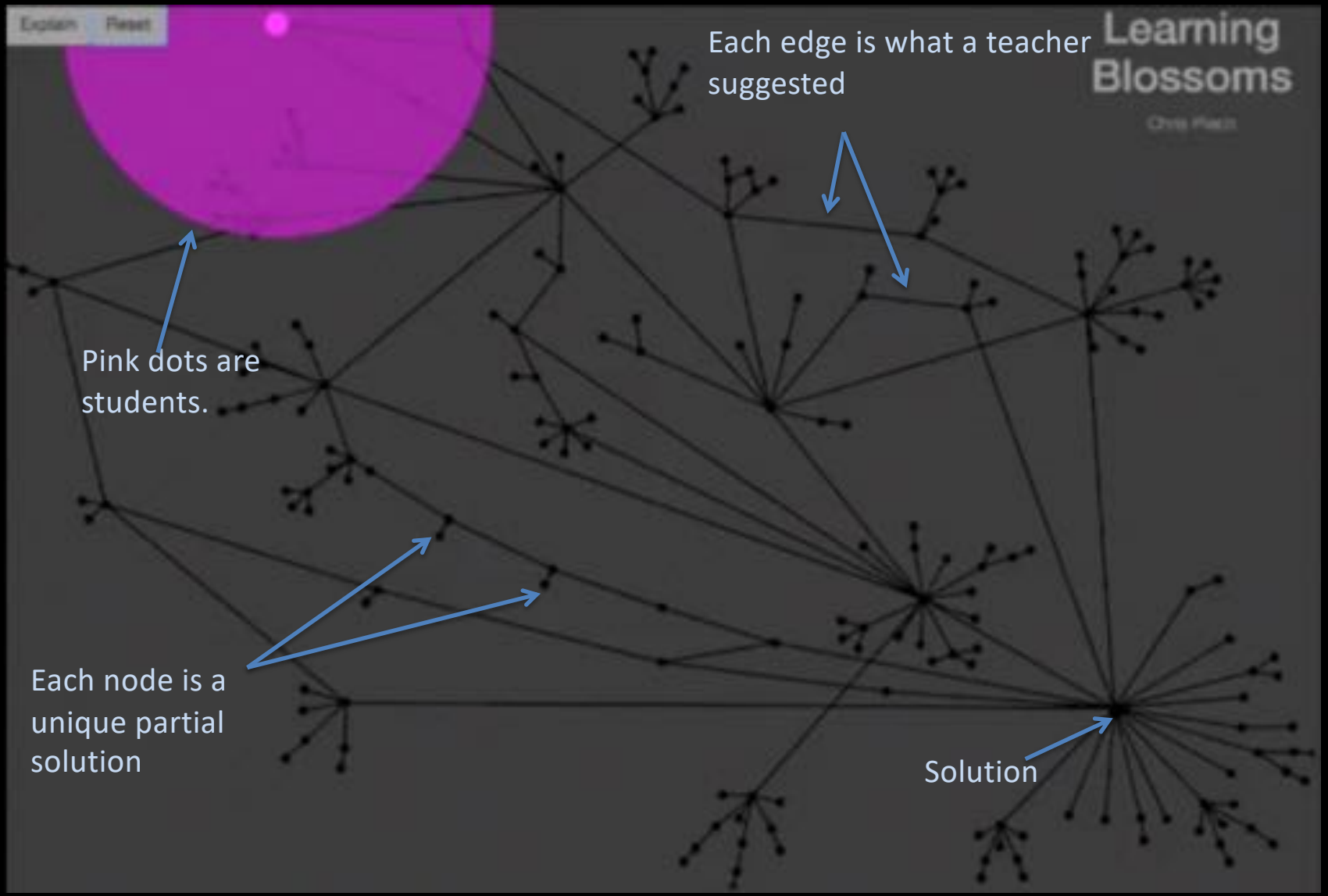
= 500,000 learners

Code.org Students

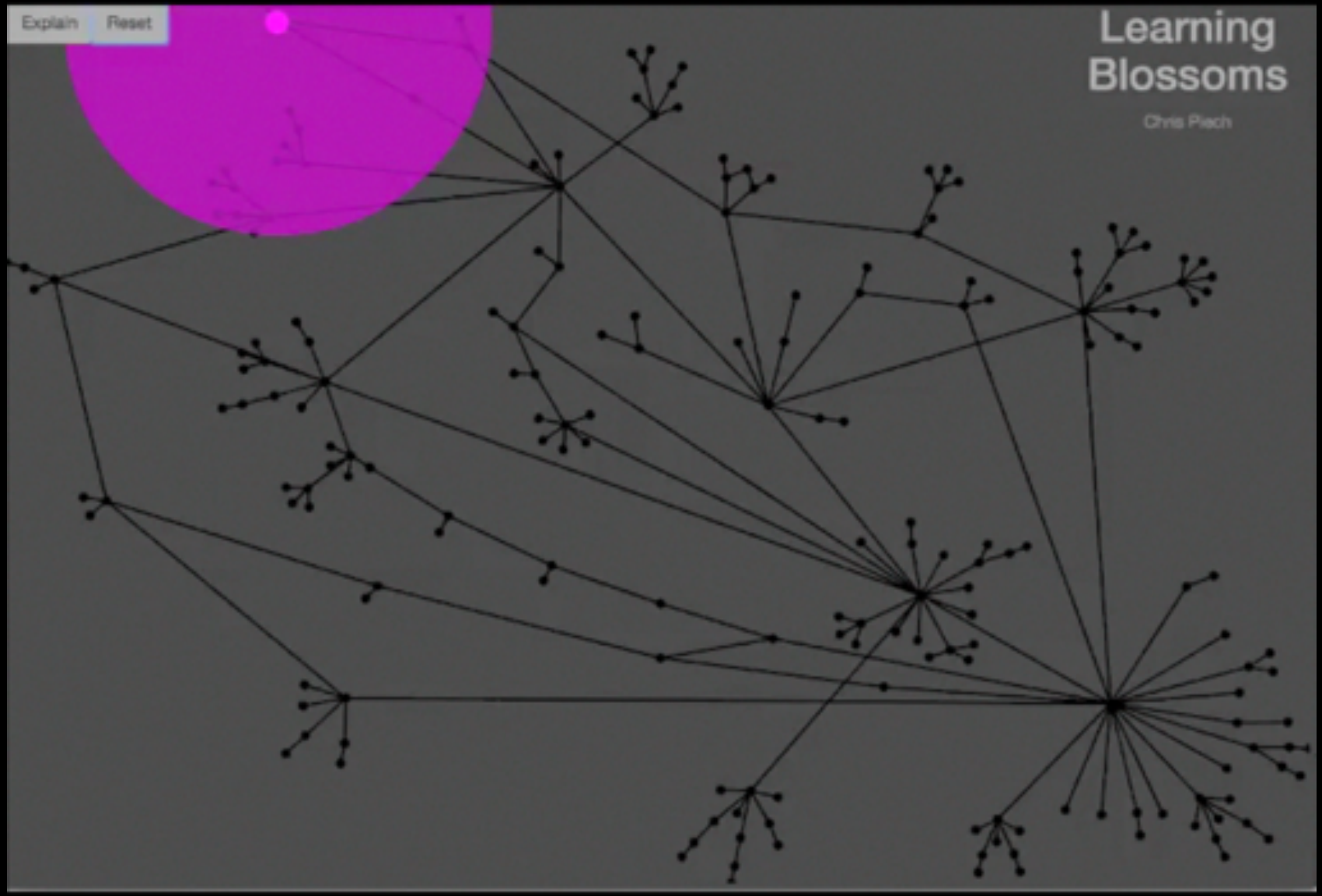




Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



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

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

Path Cost

Predicted next partial solution

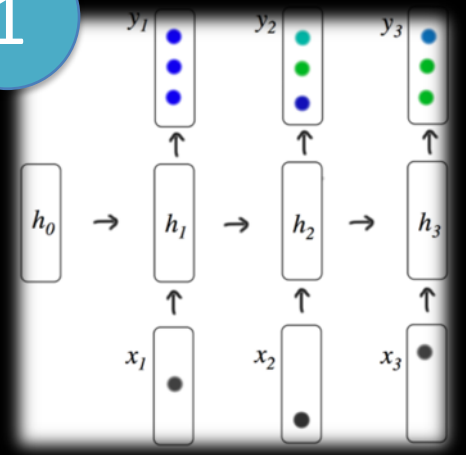
Paths to solution

Partial solutions in the path

Submission count of partial solution

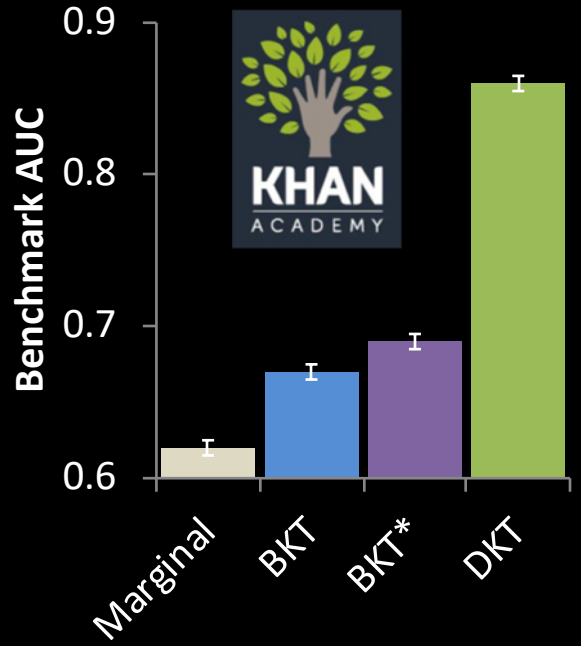
Deep Learning Algorithms

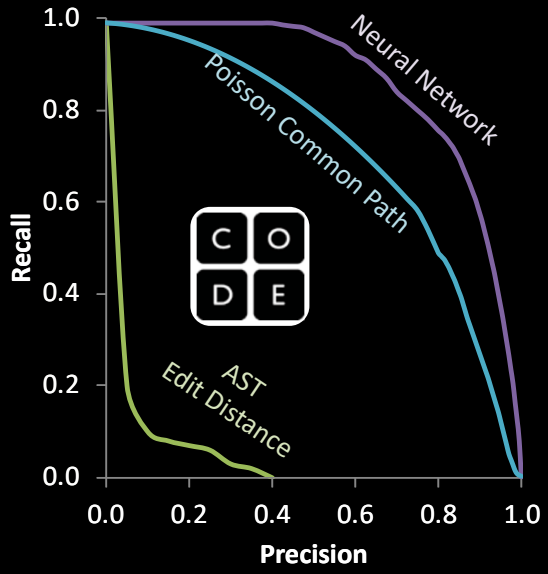
1



2

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

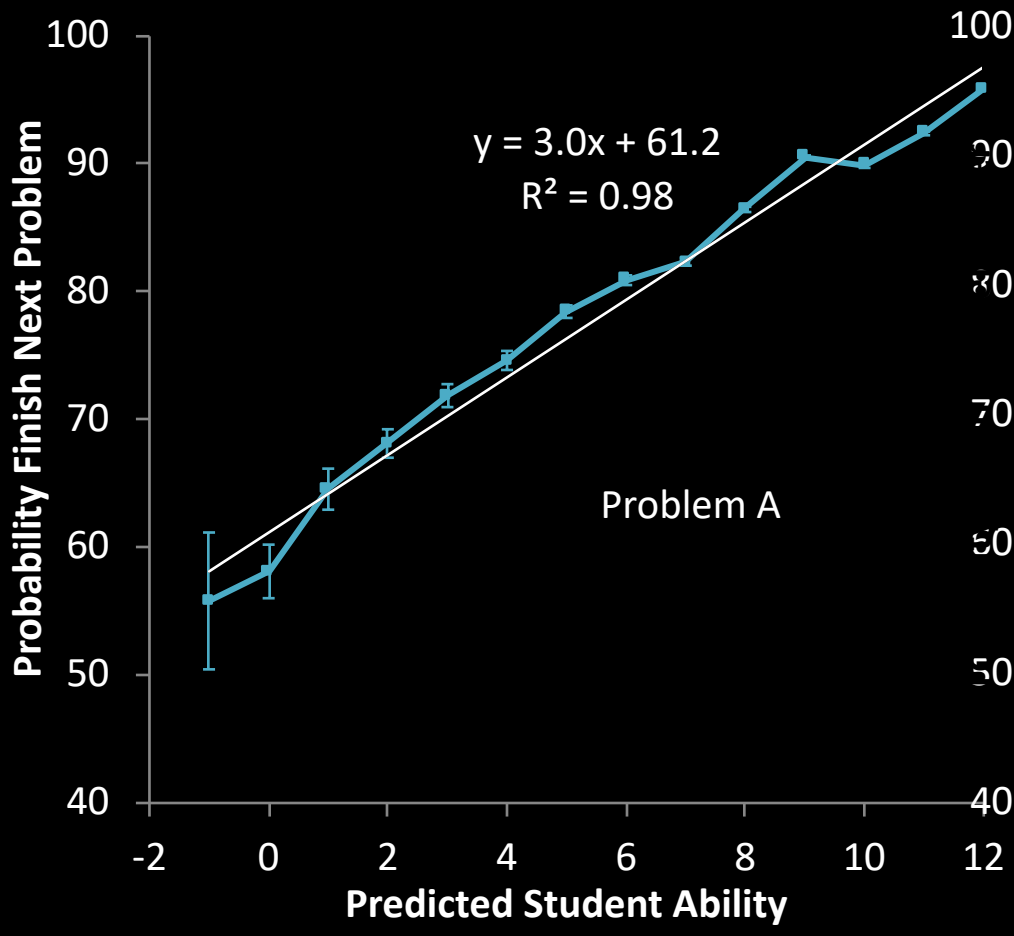


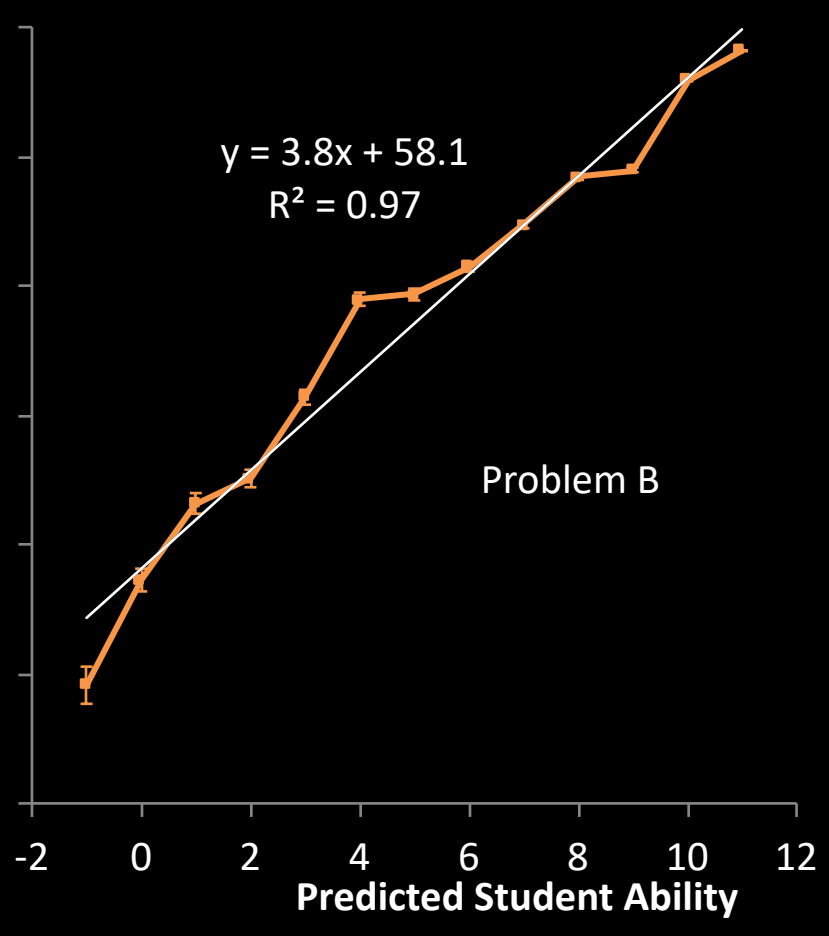


Deep Learning on Trajectories

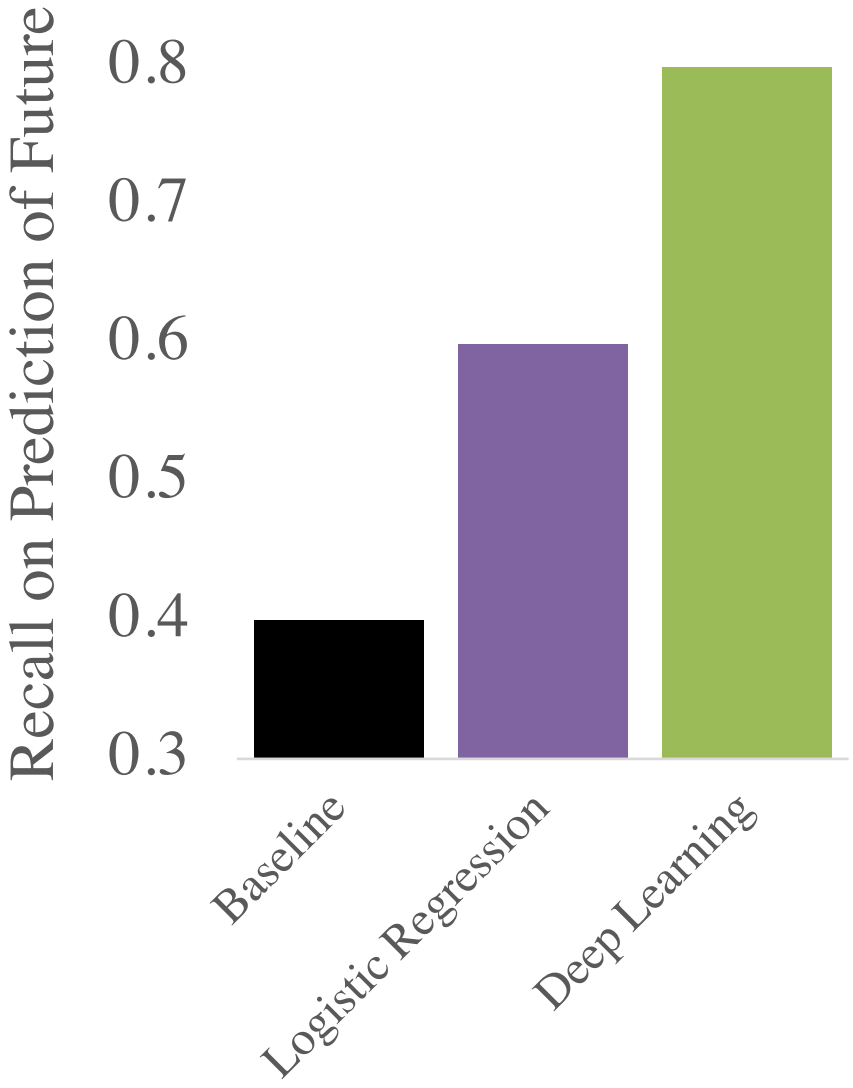
Research in collaboration with Lisa Wang

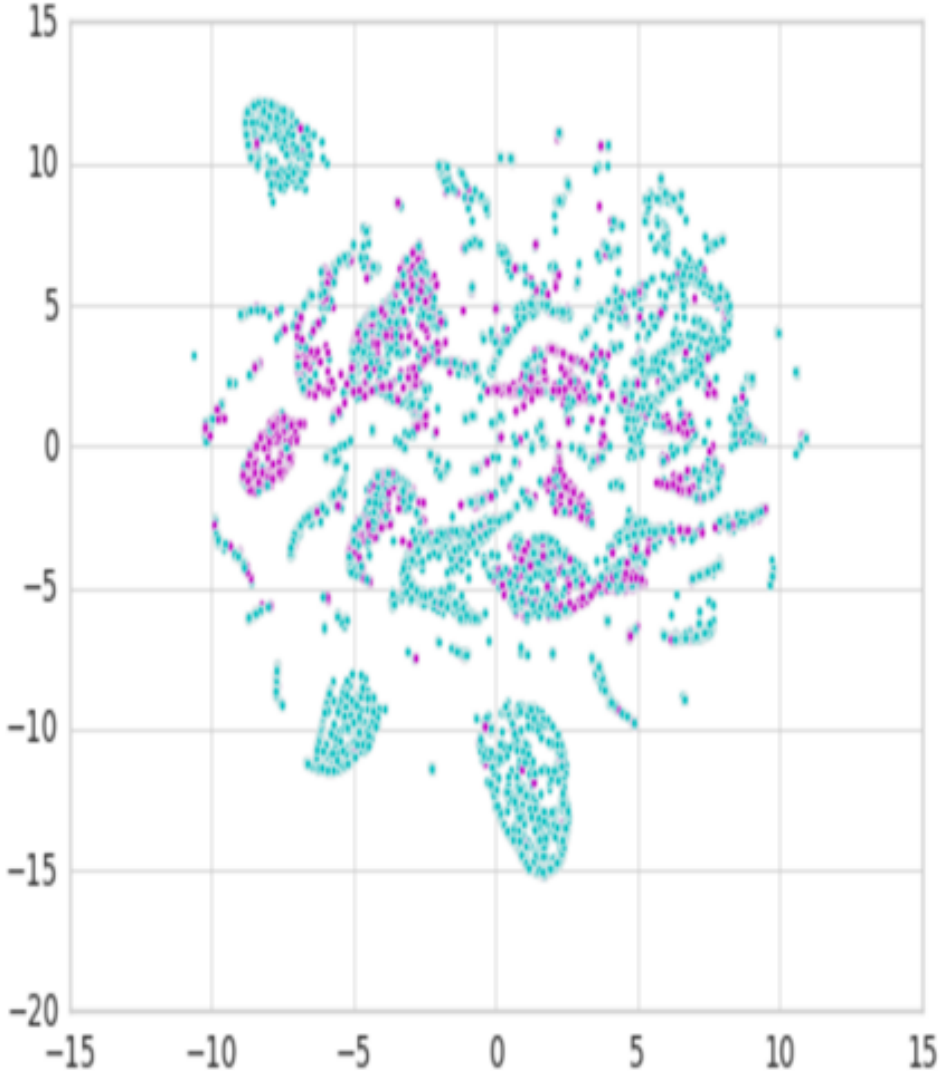
Predicts Future Success





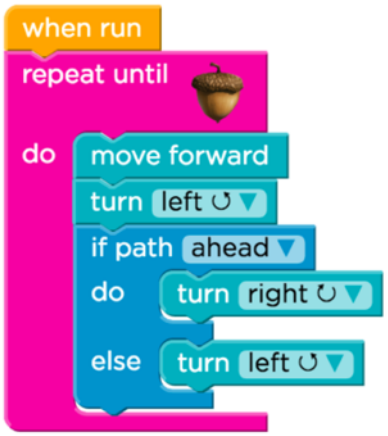
Predicts Future



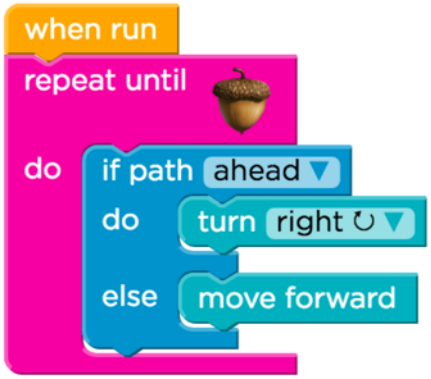


Highly Rates Grit

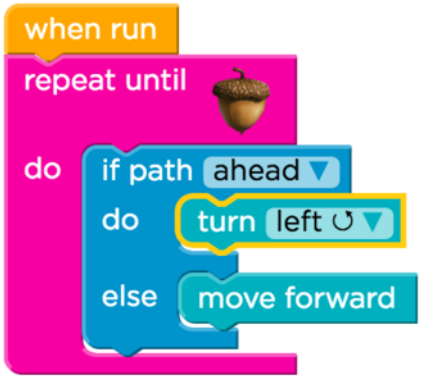
1. Two compound errors



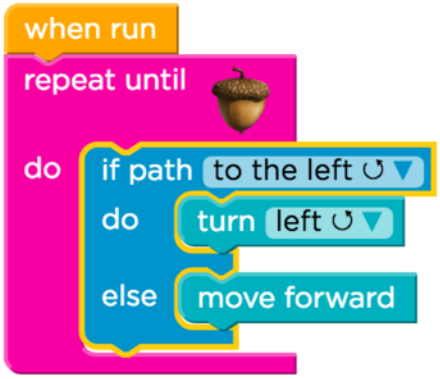
2. Solves first error



3. Starts reasonable attempt



4. Completes attempt



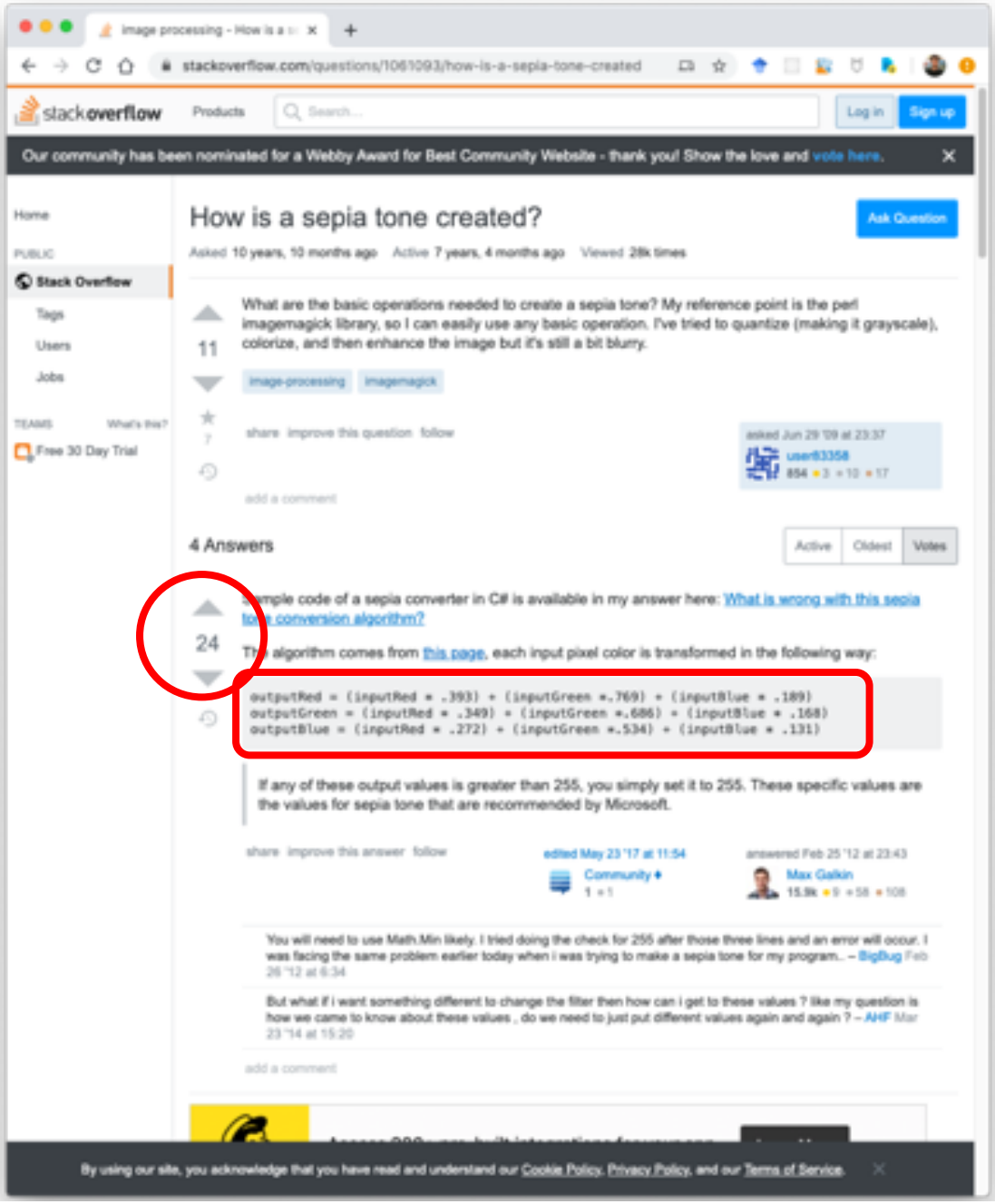
5. Backtracks

6. Finds solution

Review


CATAN


CATAN



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)
```

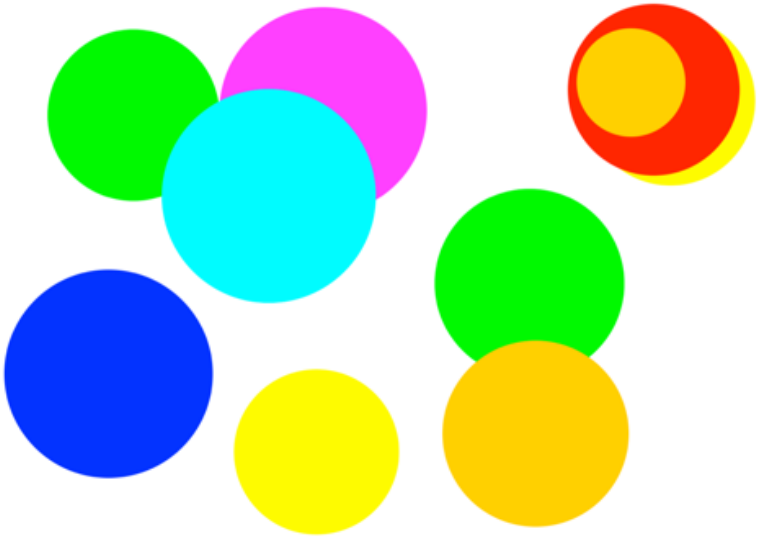

Mike Krieger **Kevin Systrom**

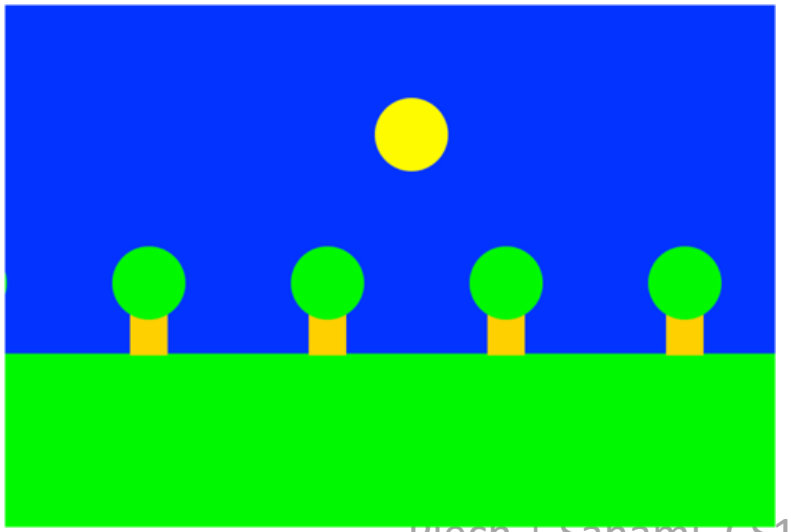
End Review

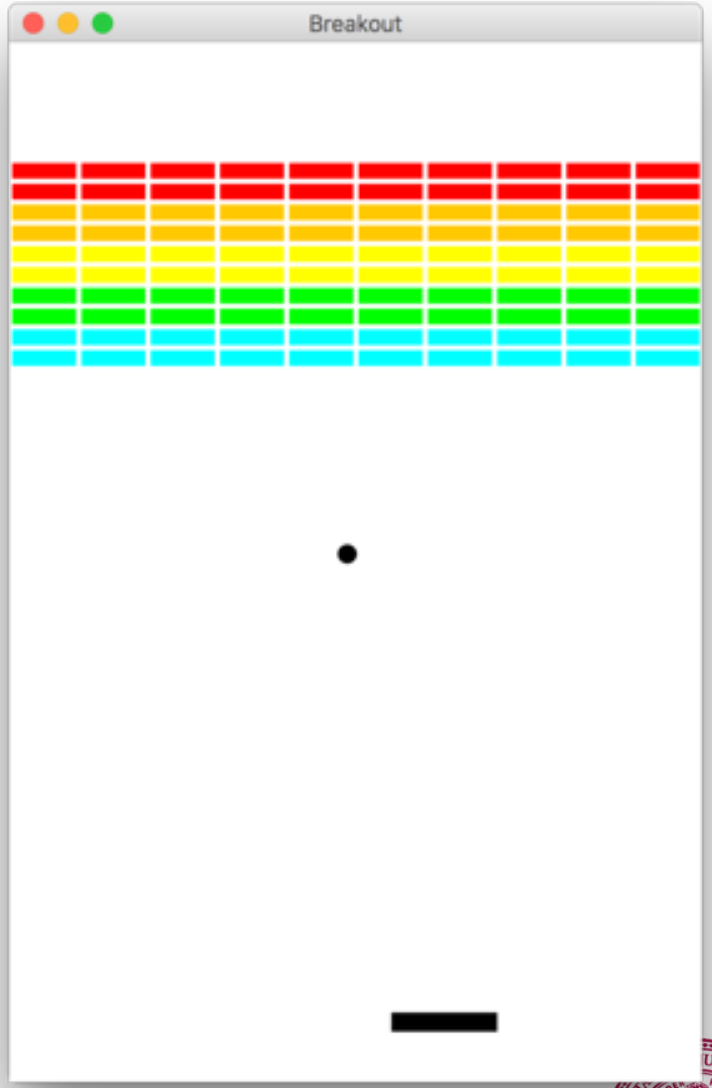
Today's Goal

1. How do I draw shapes?

Graphics Programs



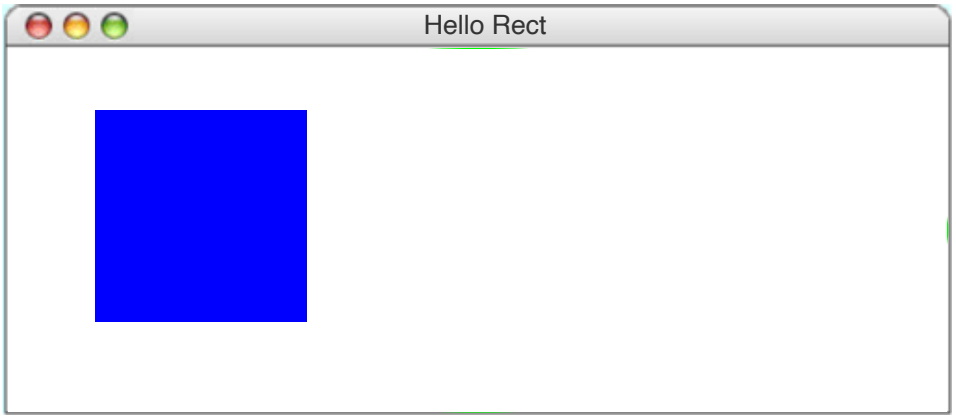




Draw a Rectangle

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

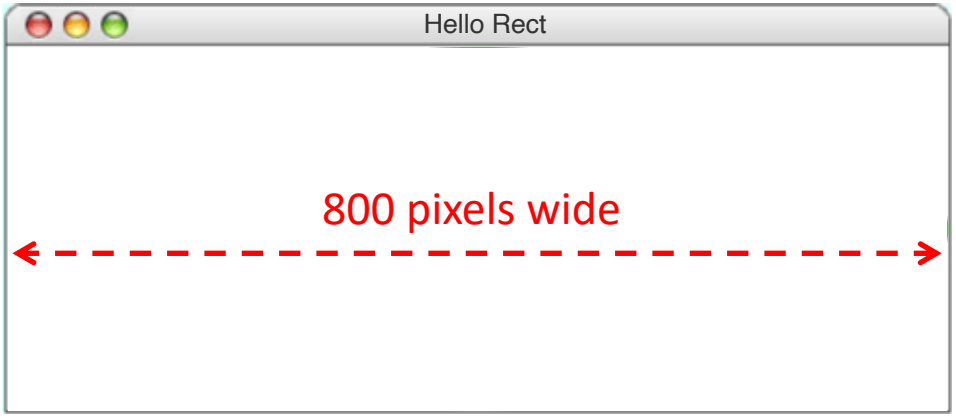
the following **`main`** method displays a blue square

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

Draw a Rectangle

the following `main` method displays a blue square

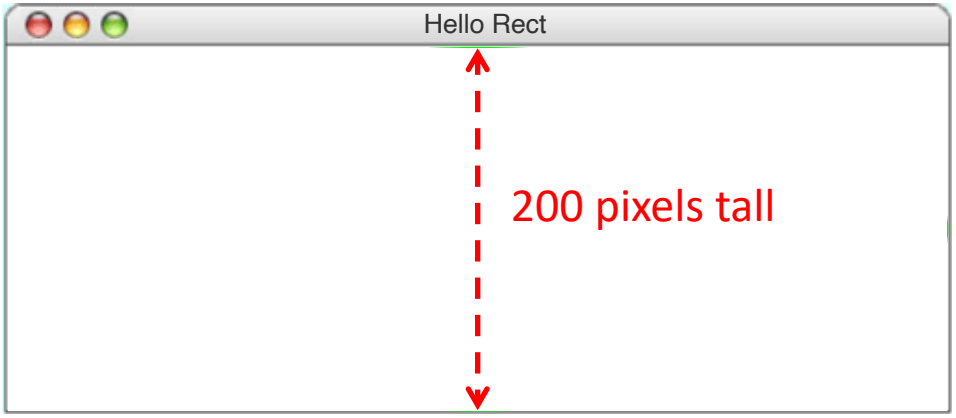
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

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



Draw a Rectangle

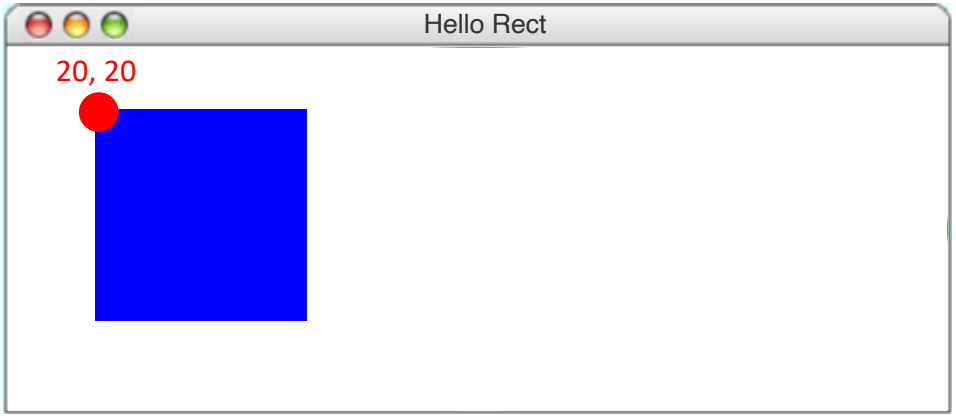
the following **`main`** method displays a blue square

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

Draw a Rectangle

the following `main` method displays a blue square

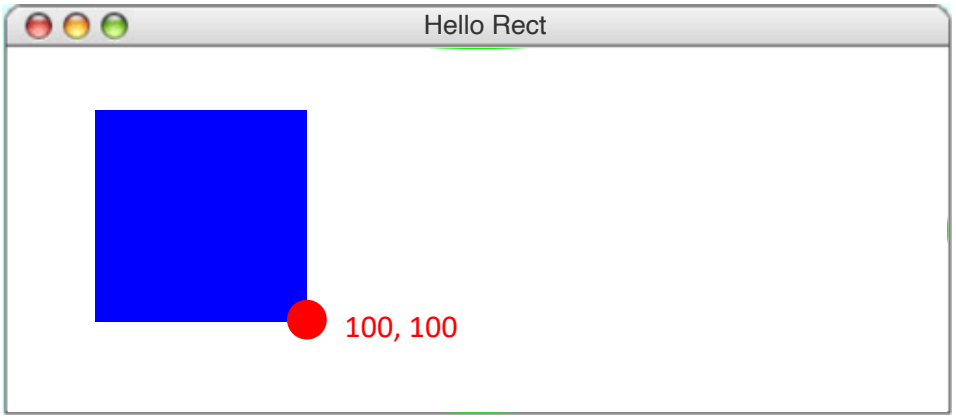
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

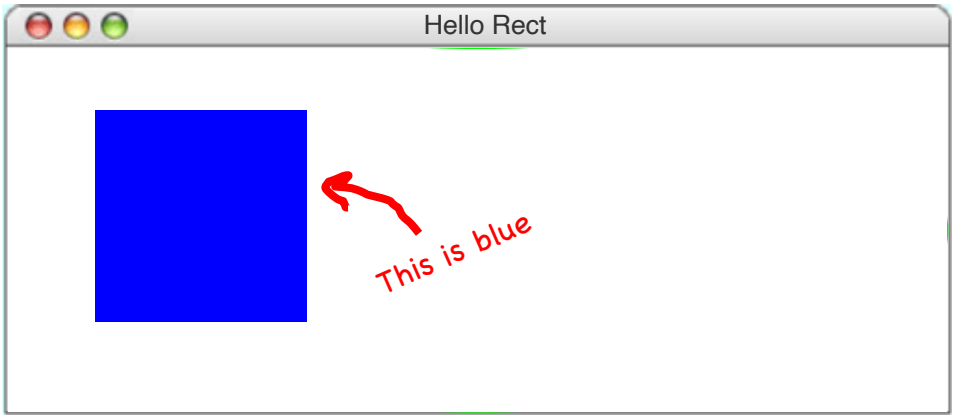
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



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")
```

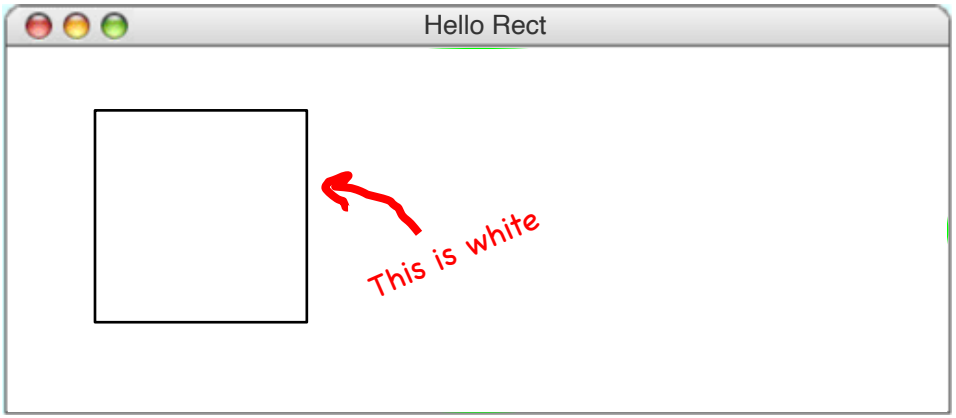


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

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)
```

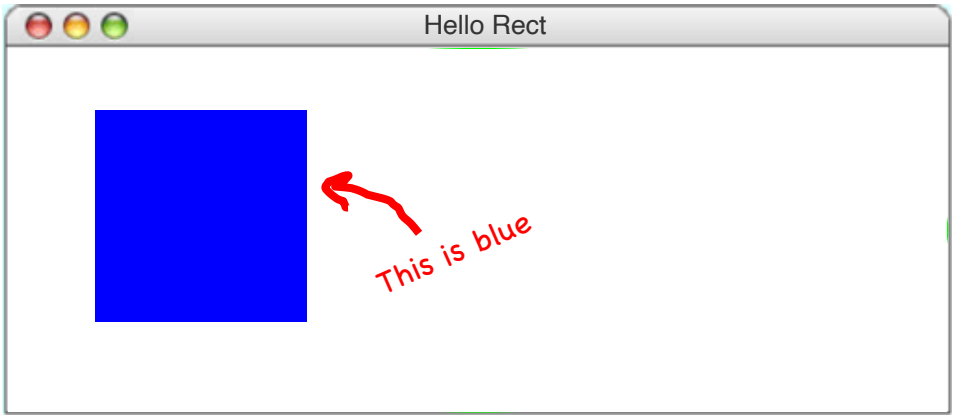


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

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")
```

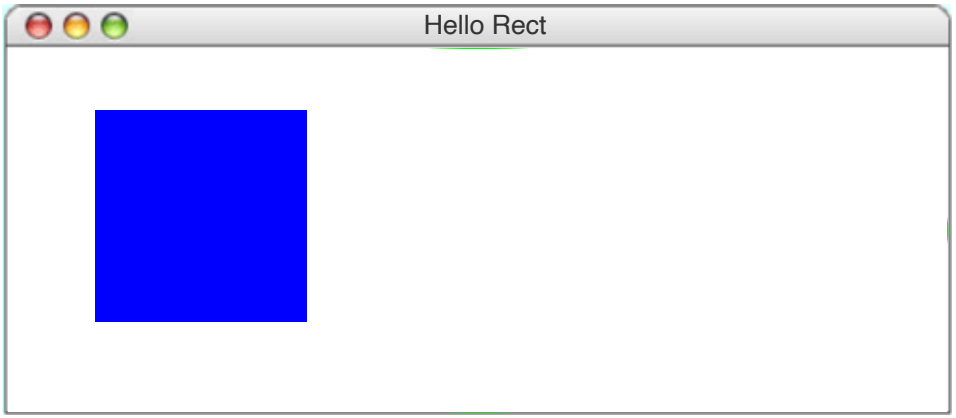


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

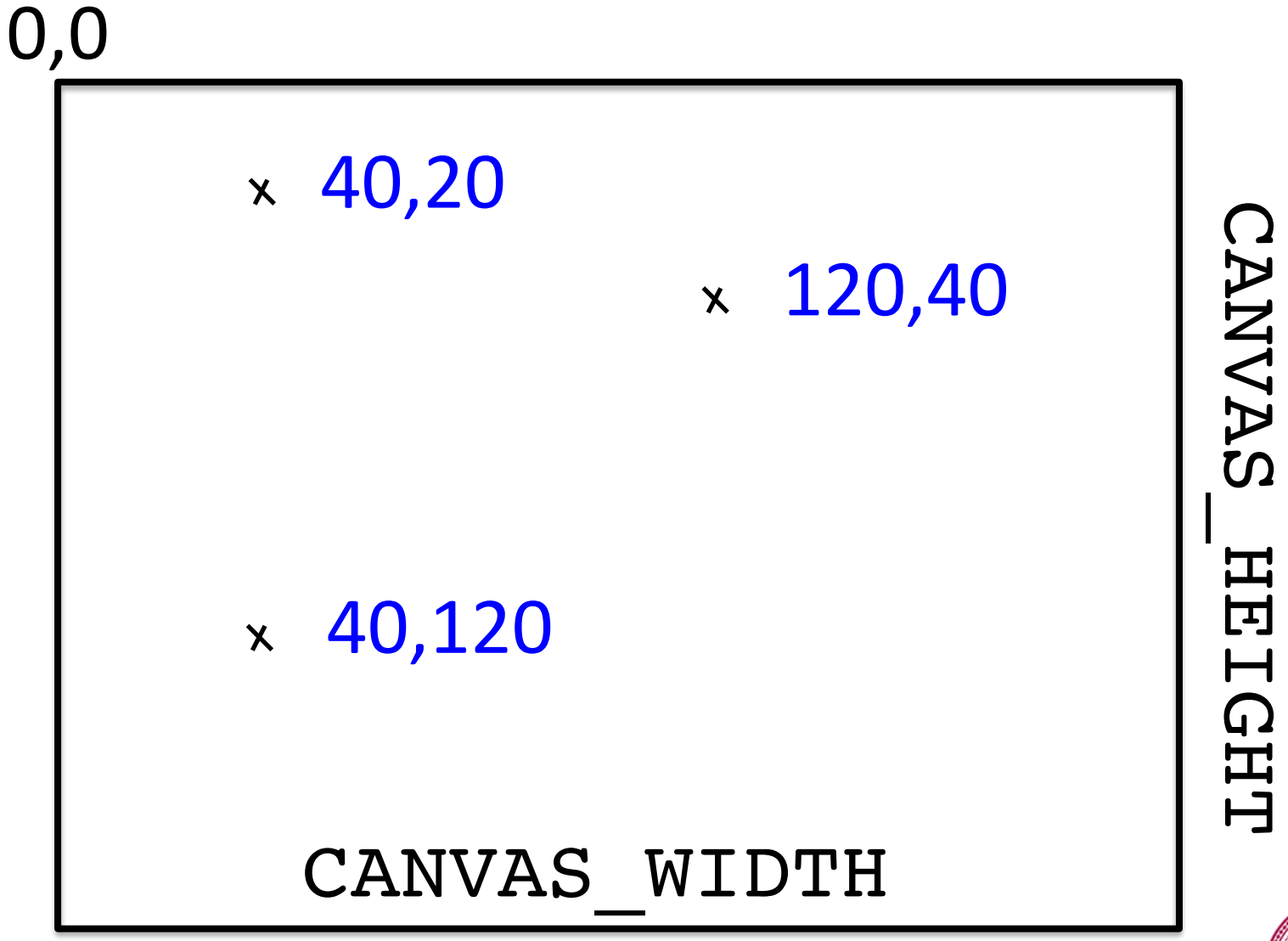
the following **main** method displays a blue square

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

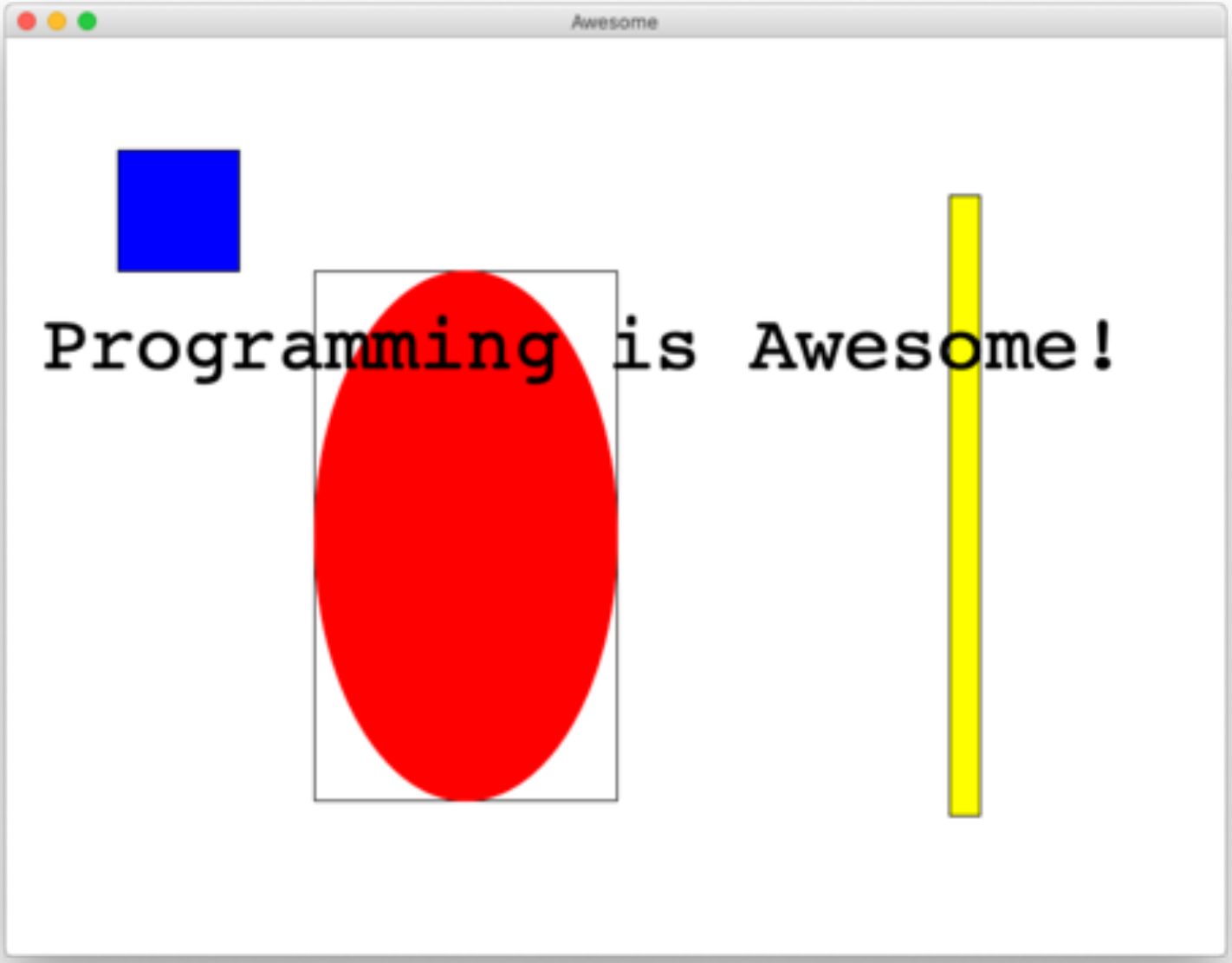


TK Natural Graphics

Graphics Coordinates



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)`

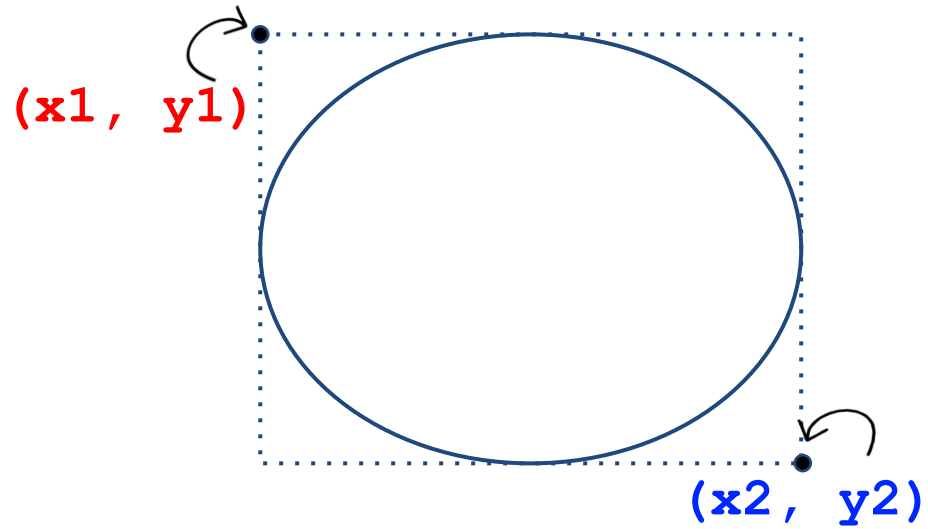
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

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

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

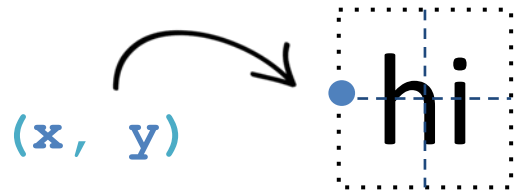
- `canvas.create_line()`
- **`canvas.create_oval(x1, y1, x2, y2)`**
- `canvas.create_text()`

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



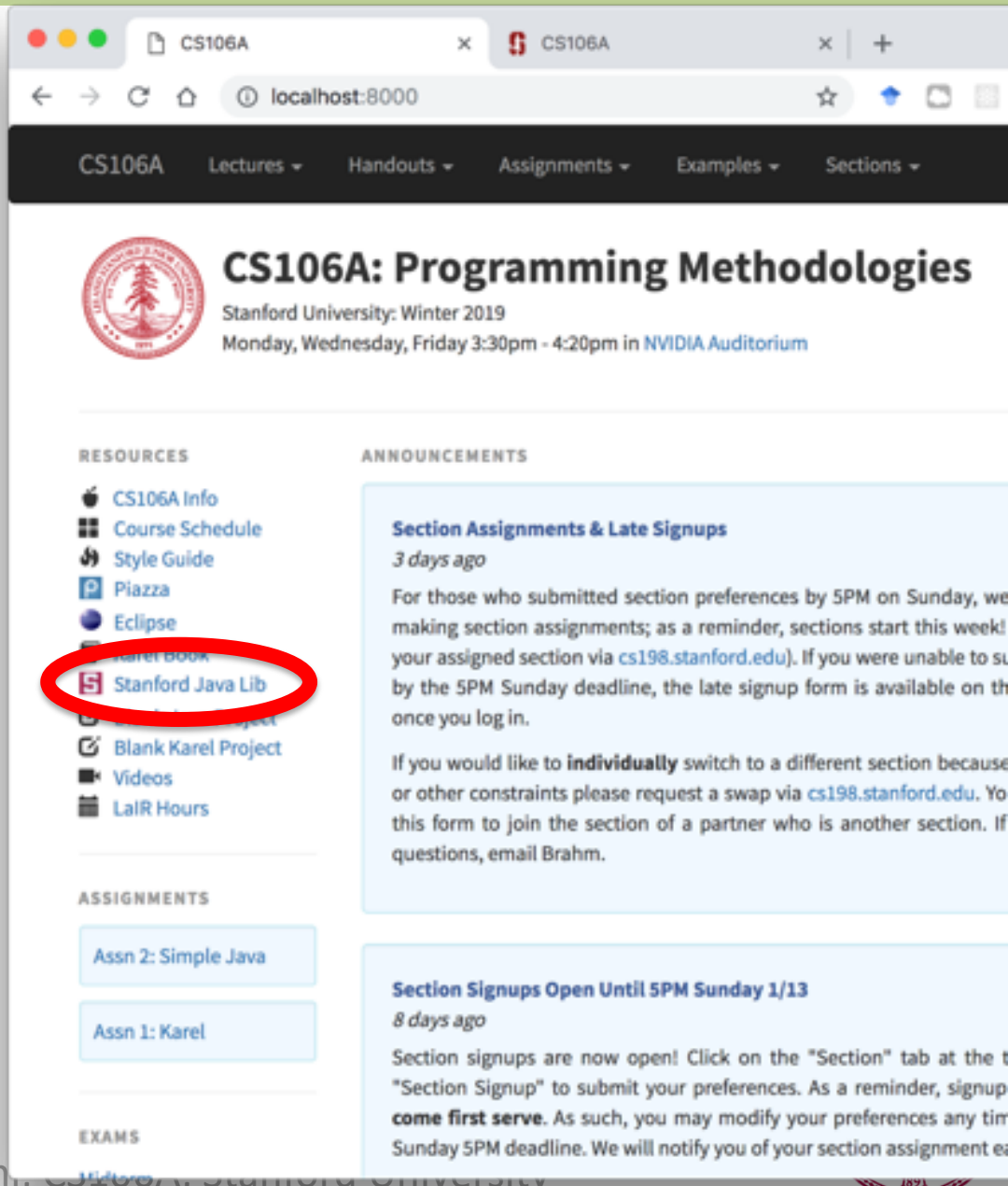
- `canvas.create_line()`
- `canvas.create_oval()`
- **`canvas.create_text(x, y, text='hi')`**

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

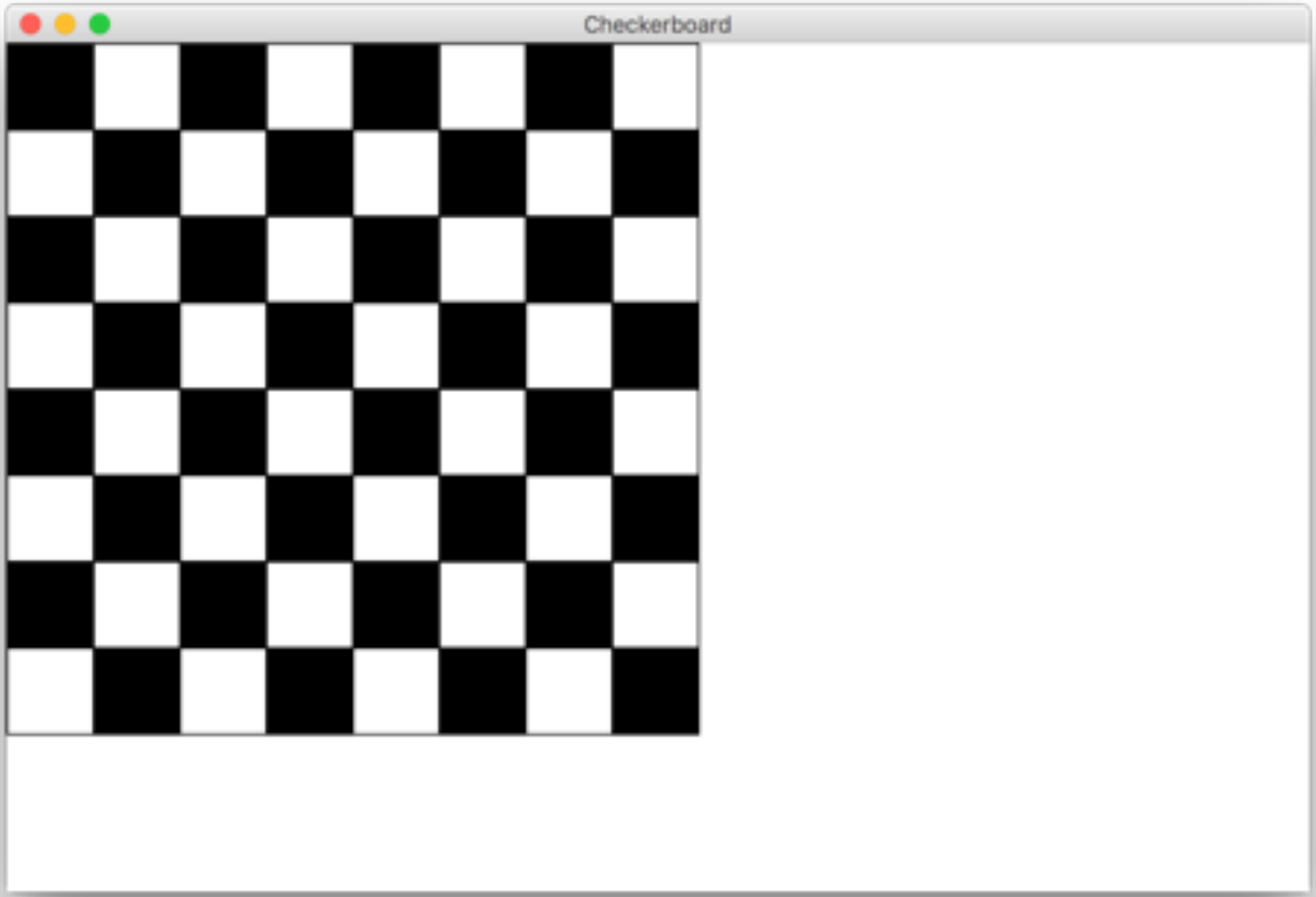


Pedagogy

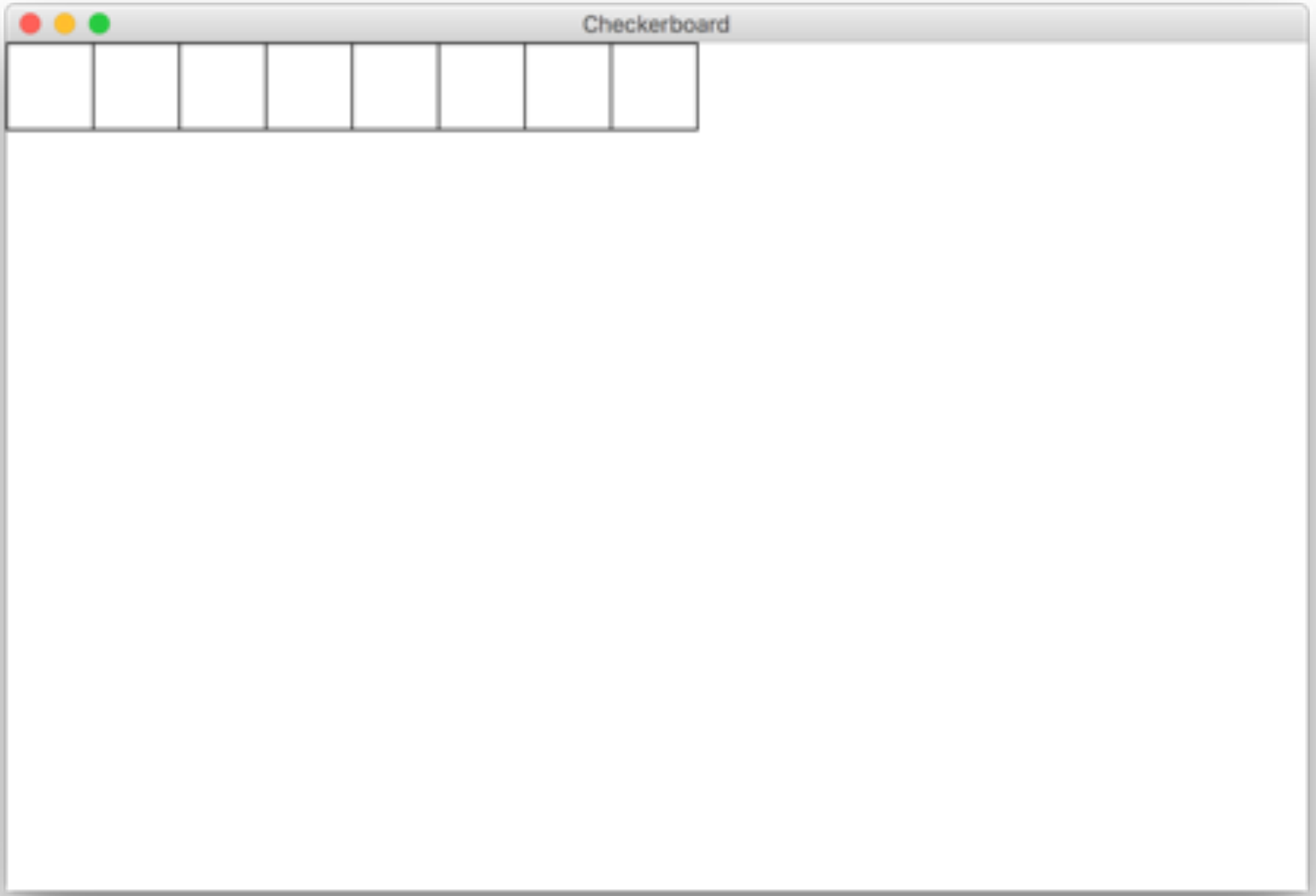

CS106A
localhost:8000/examples/...
Programming is Awesome
Programming is Awesome!


CS106A
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
LaIR Hours
ASSIGNMENTS
Assn 2: Simple Java
Assn 1: Karel
EXAMS
ANNOUNCEMENTS
Section Assignments & Late Signups
3 days ago
Section Signups Open Until 5PM Sunday 1/13
8 days ago

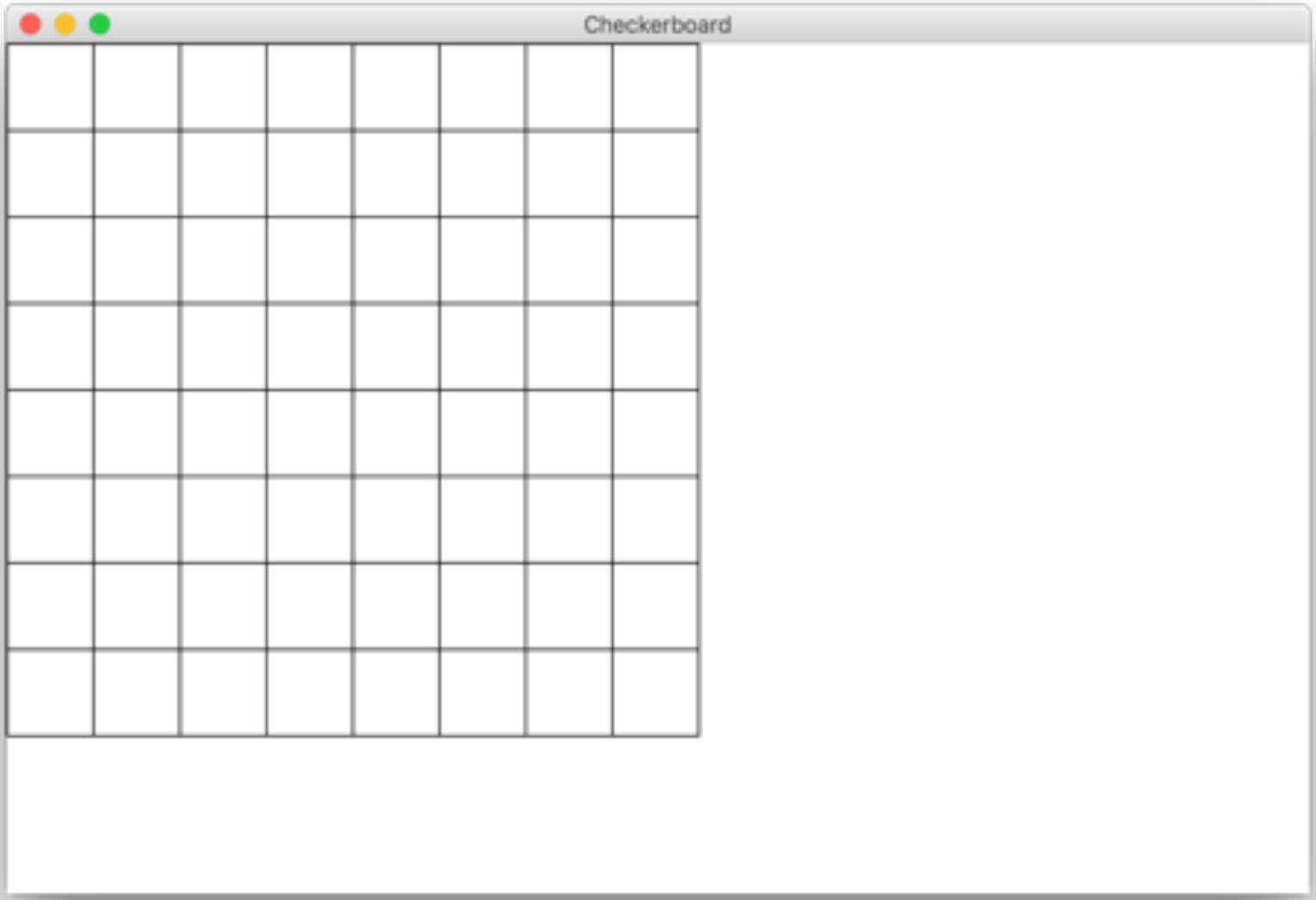
Goal



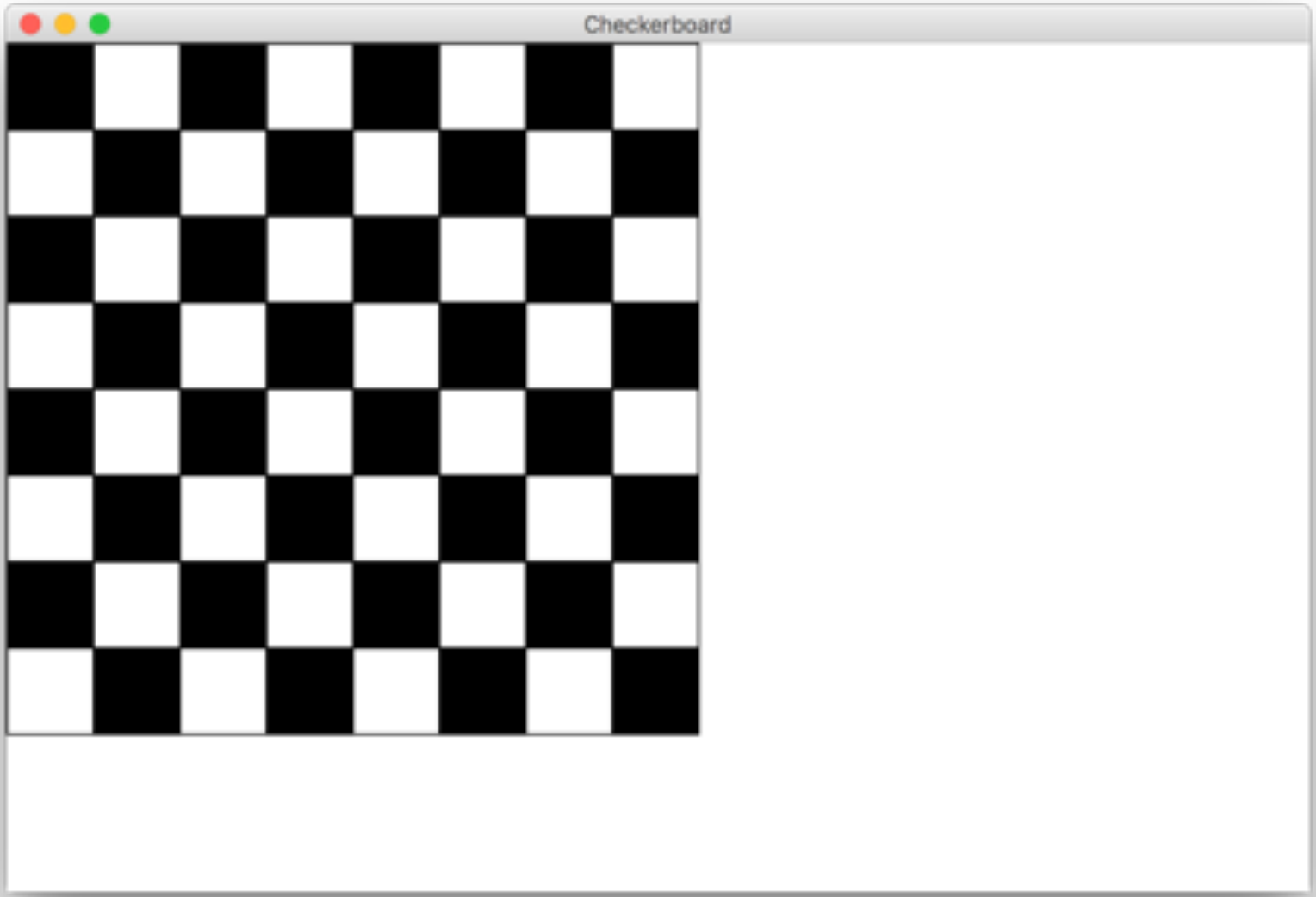
Milestone 1

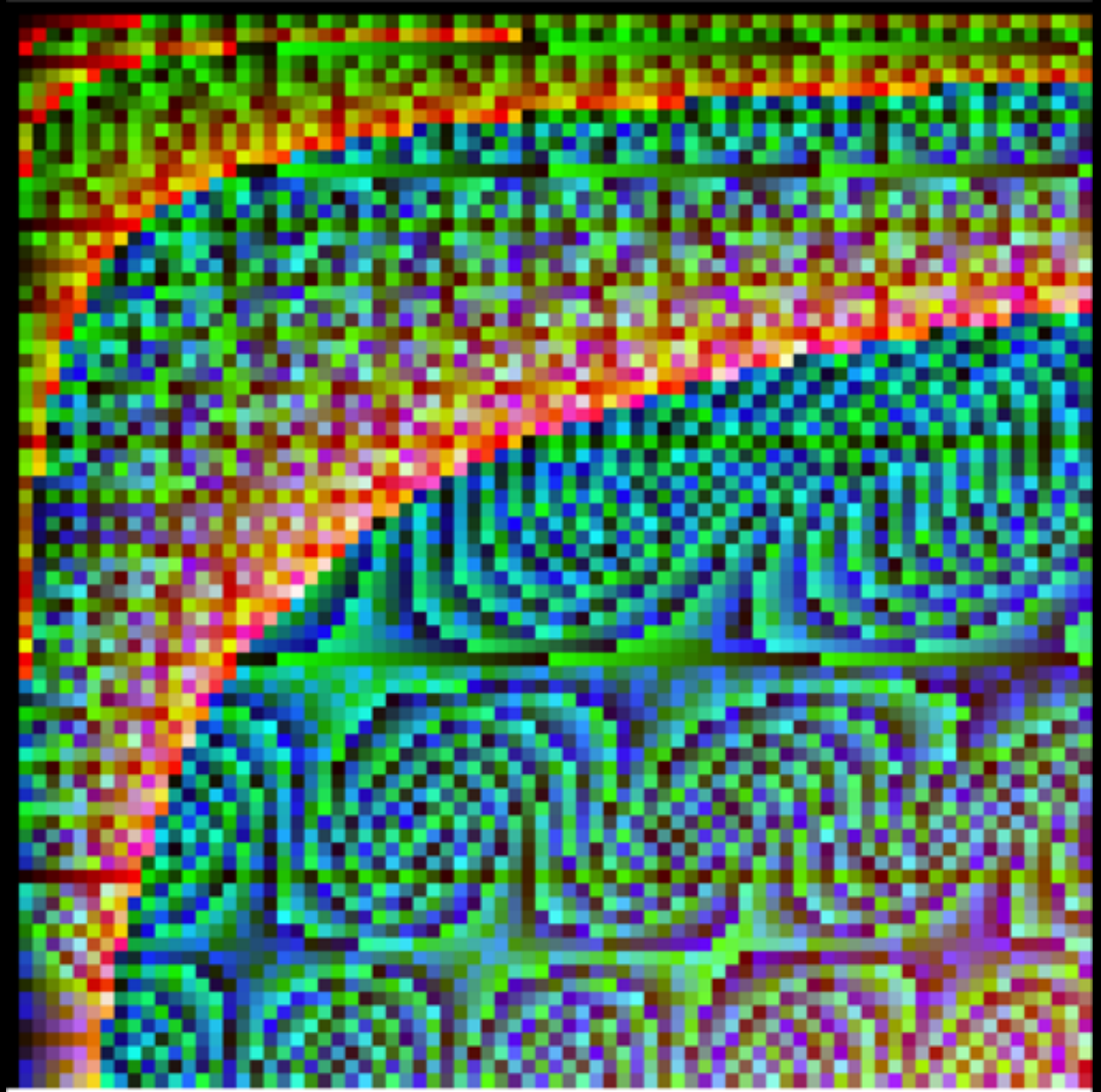


Milestone 2



Milestone 3





Teaser for next week…

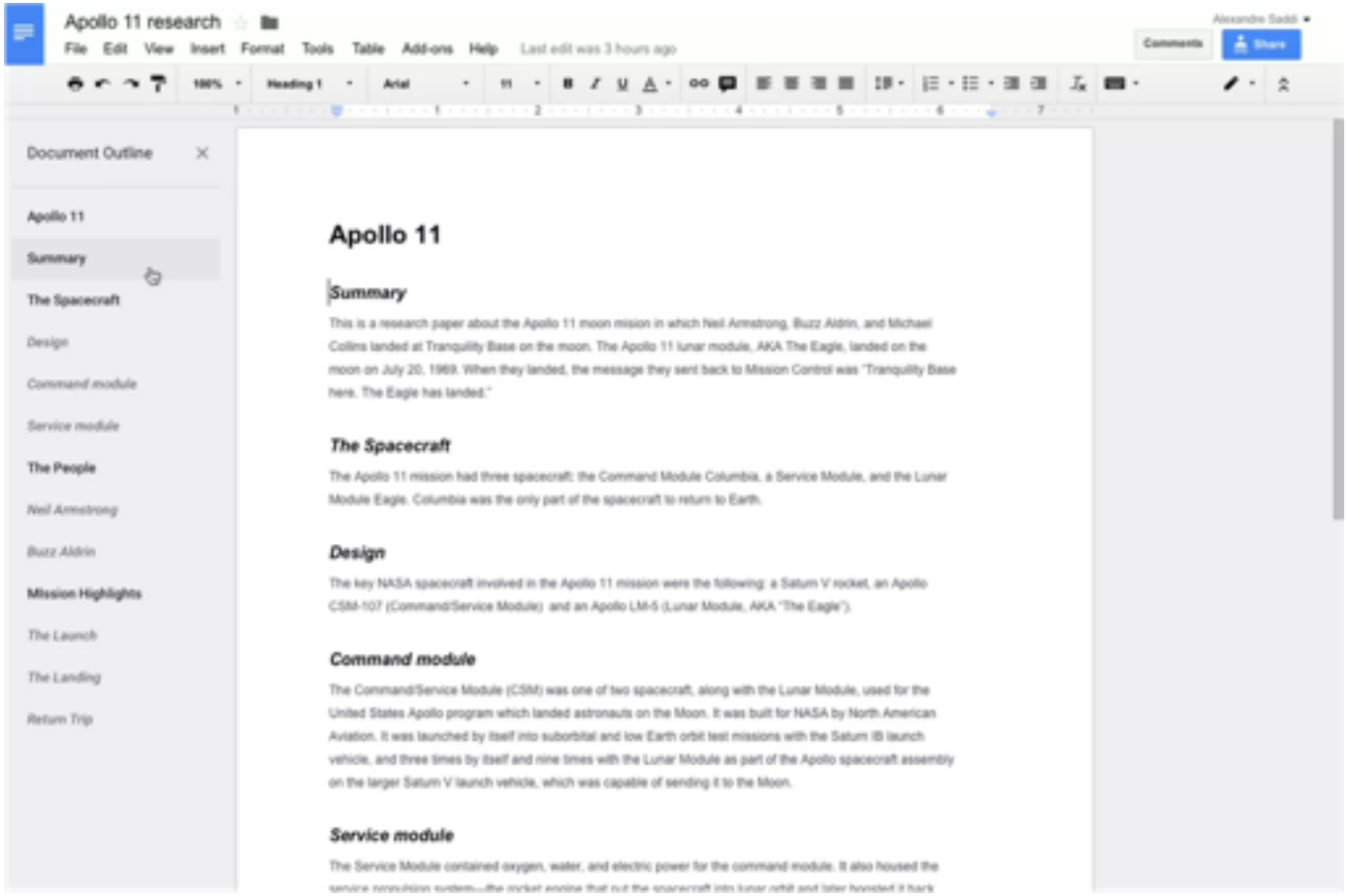
Hold up!

```
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/1eBtnEiII3KHefFS-kSAOpXqeSXpbfTTMlmOgj6I9dvk/

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

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

---

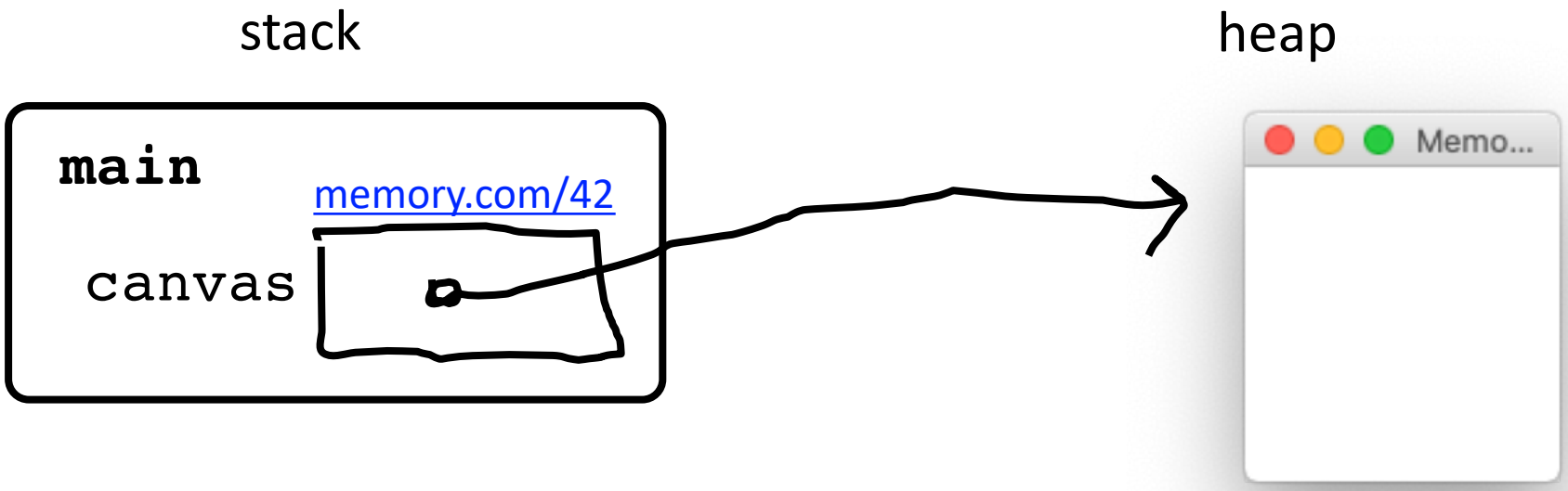
stack

heap

main

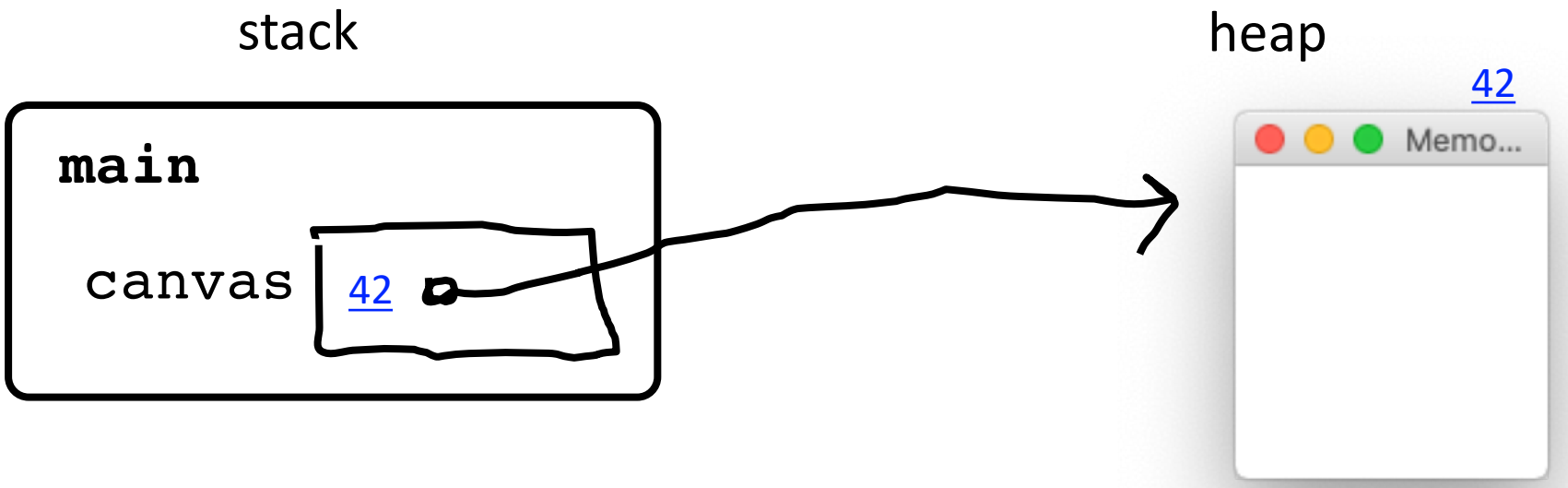
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



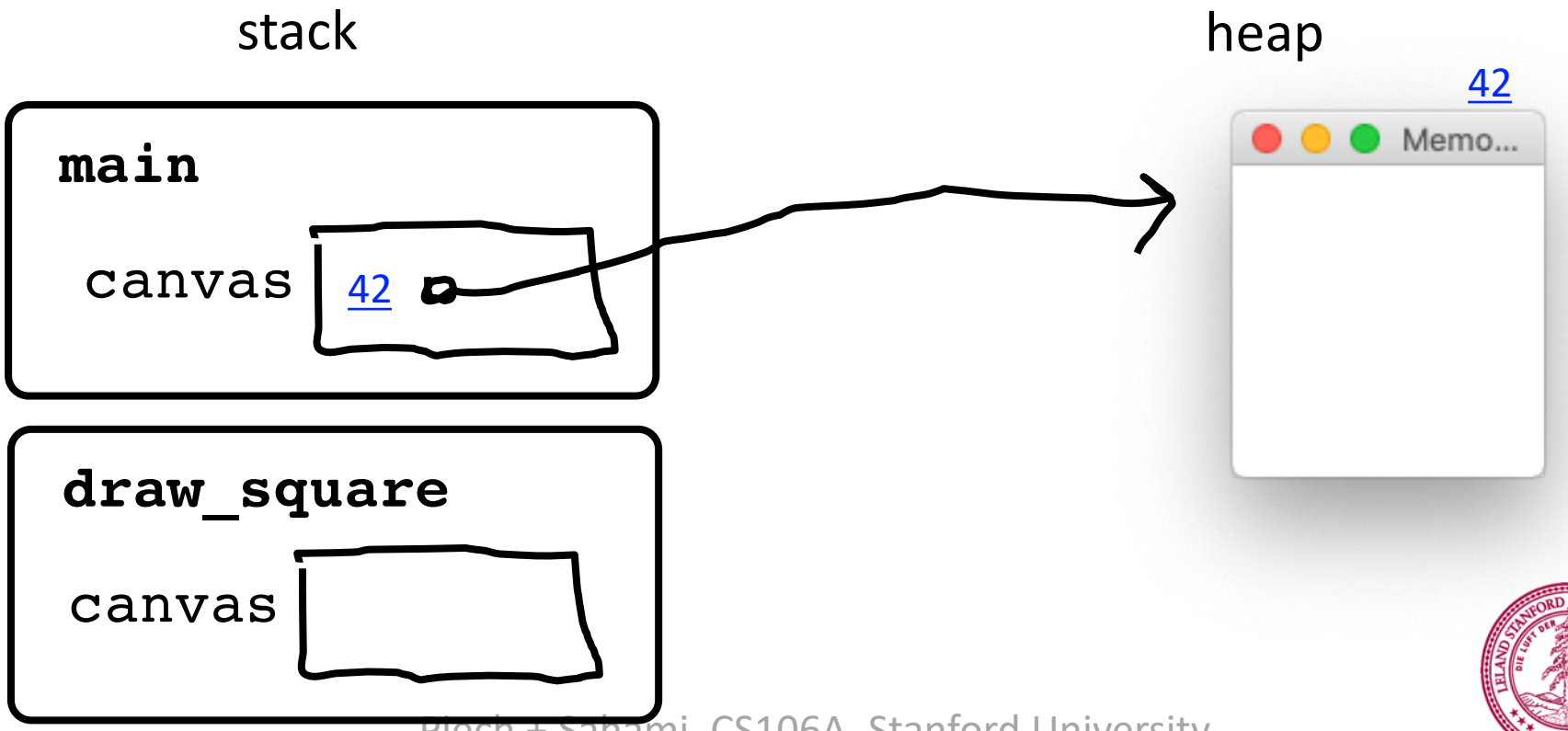
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



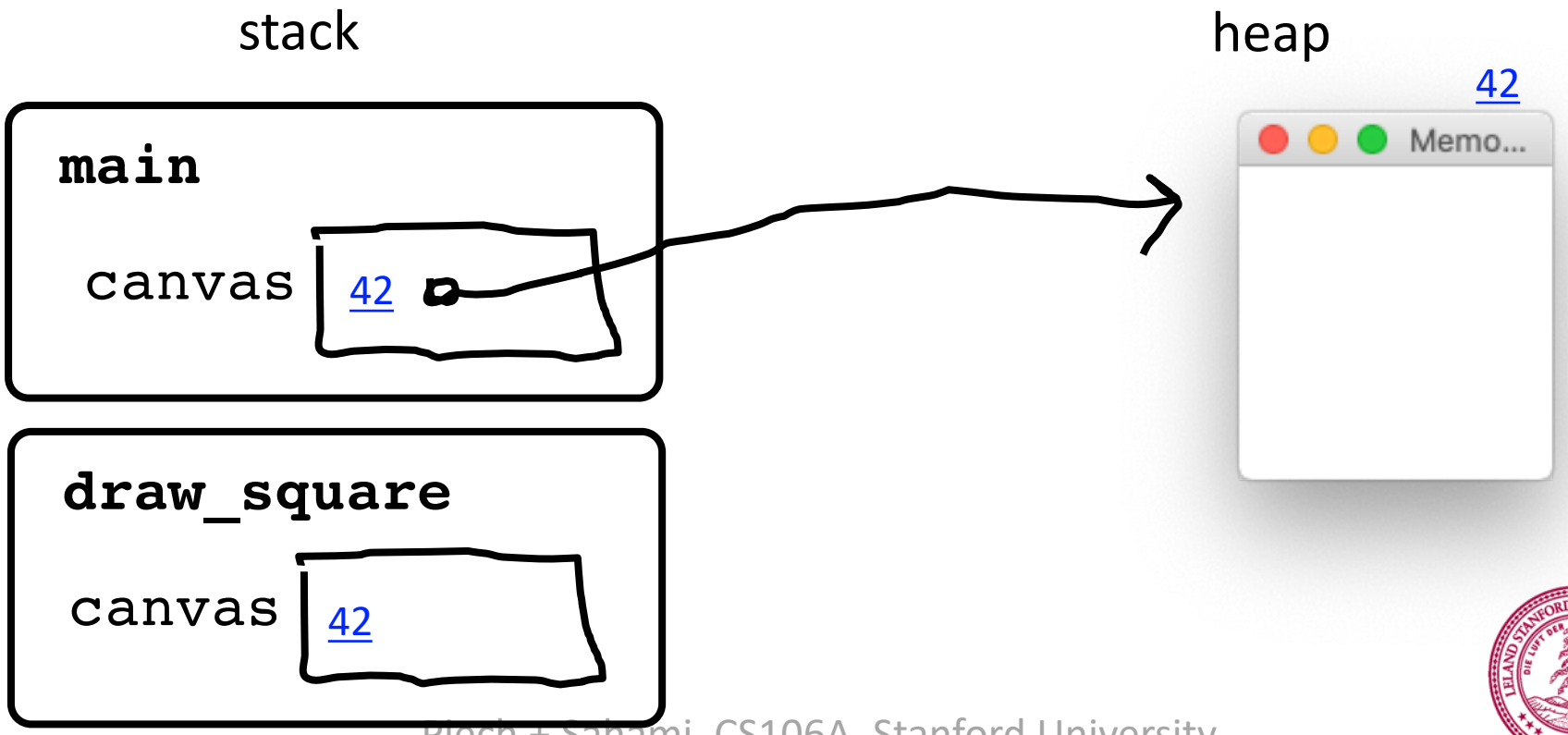
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



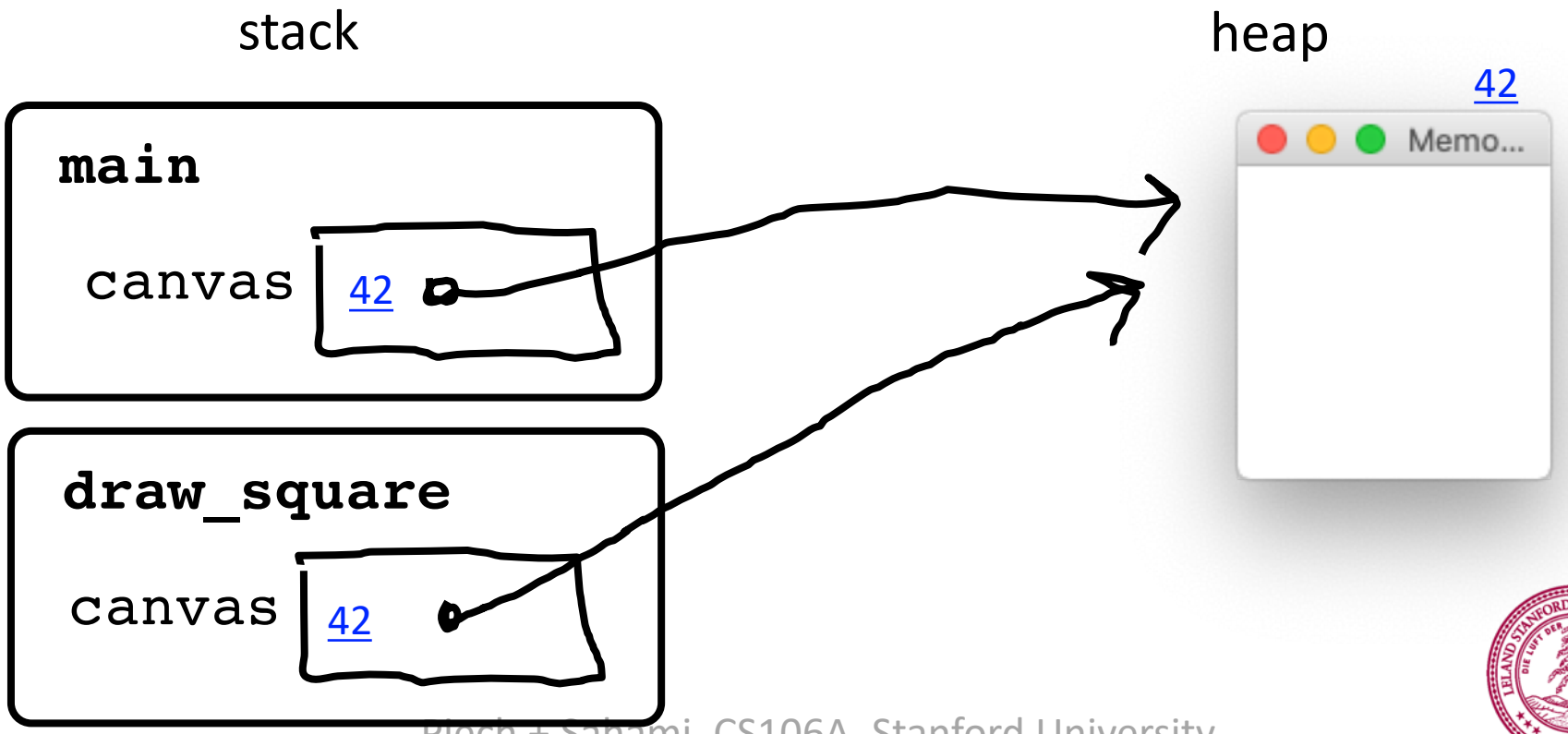
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



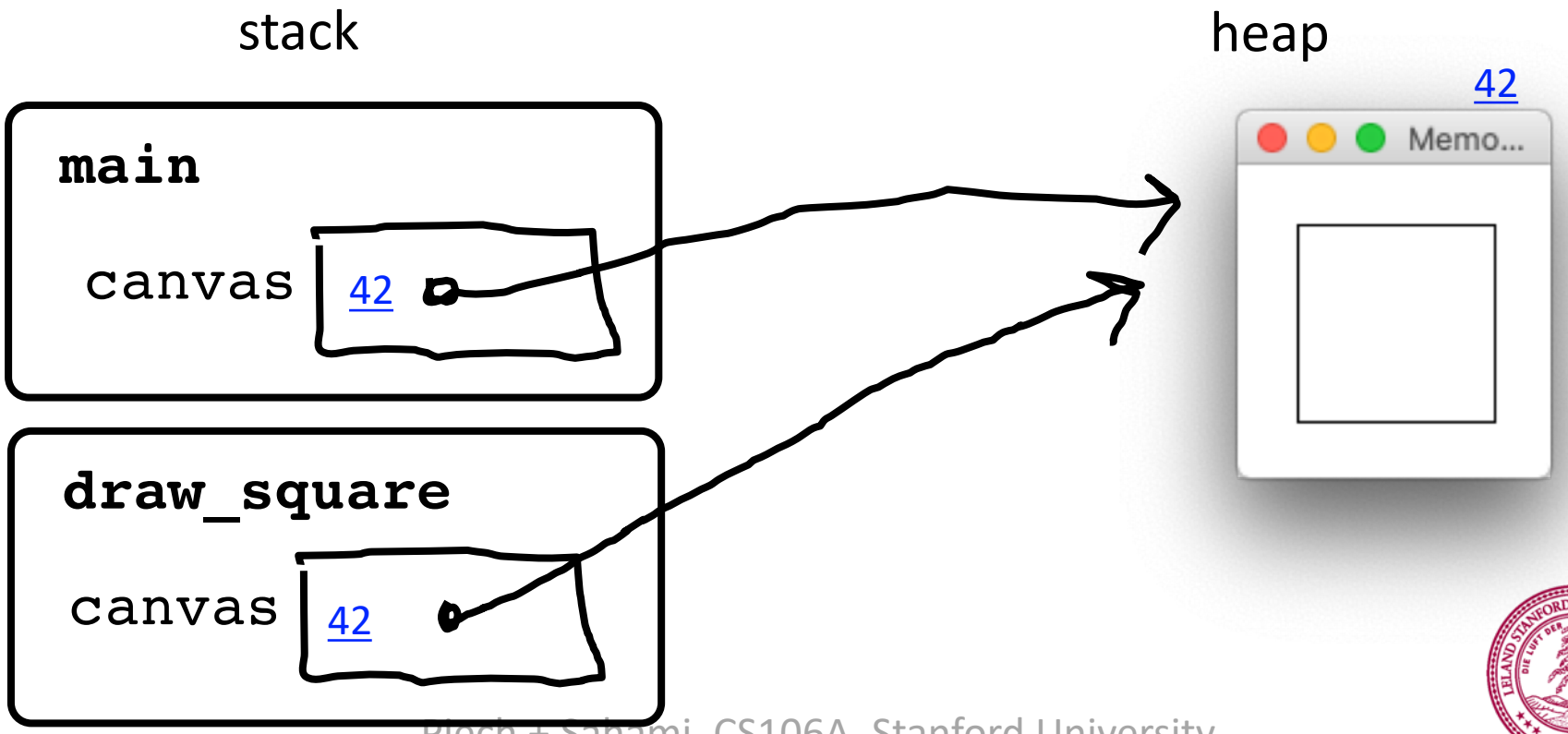
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



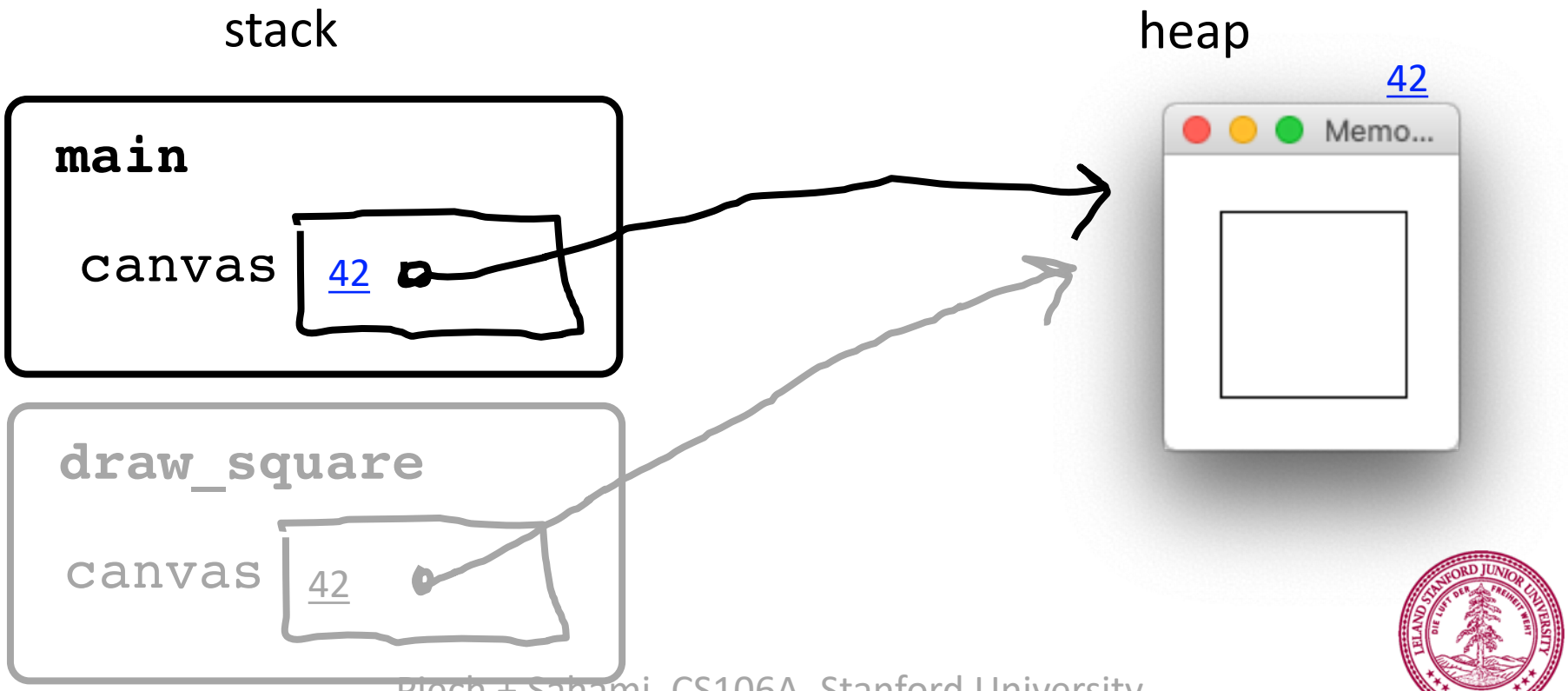
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



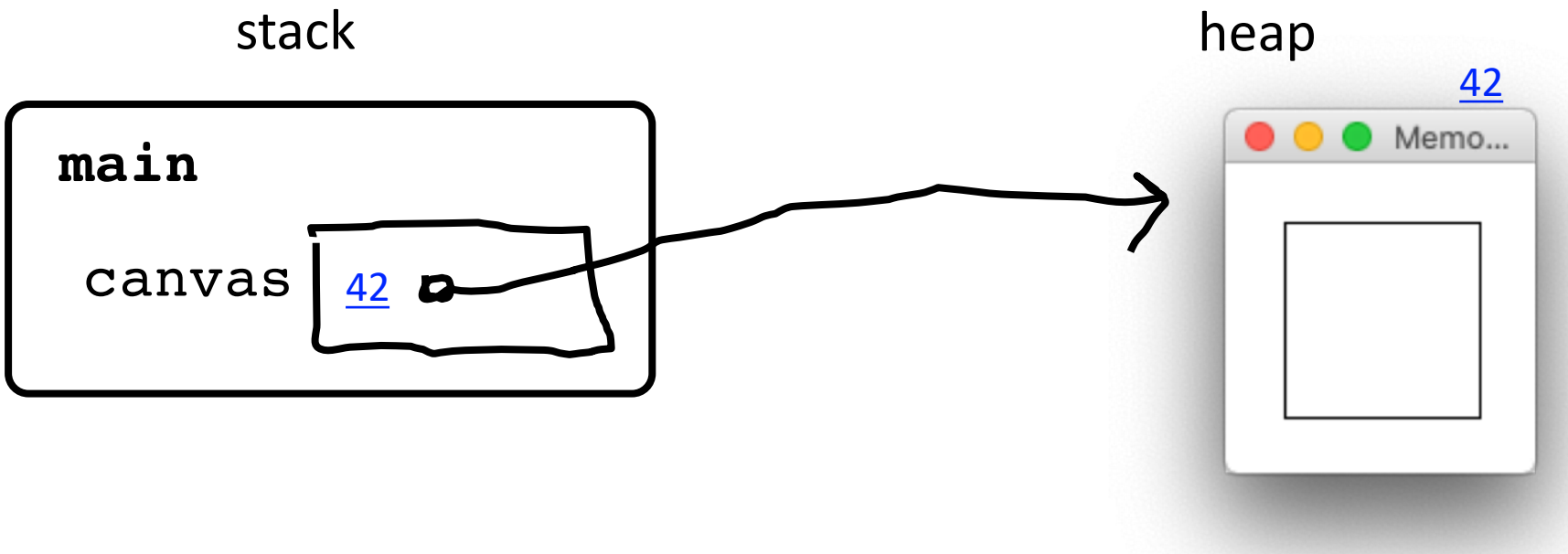
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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)