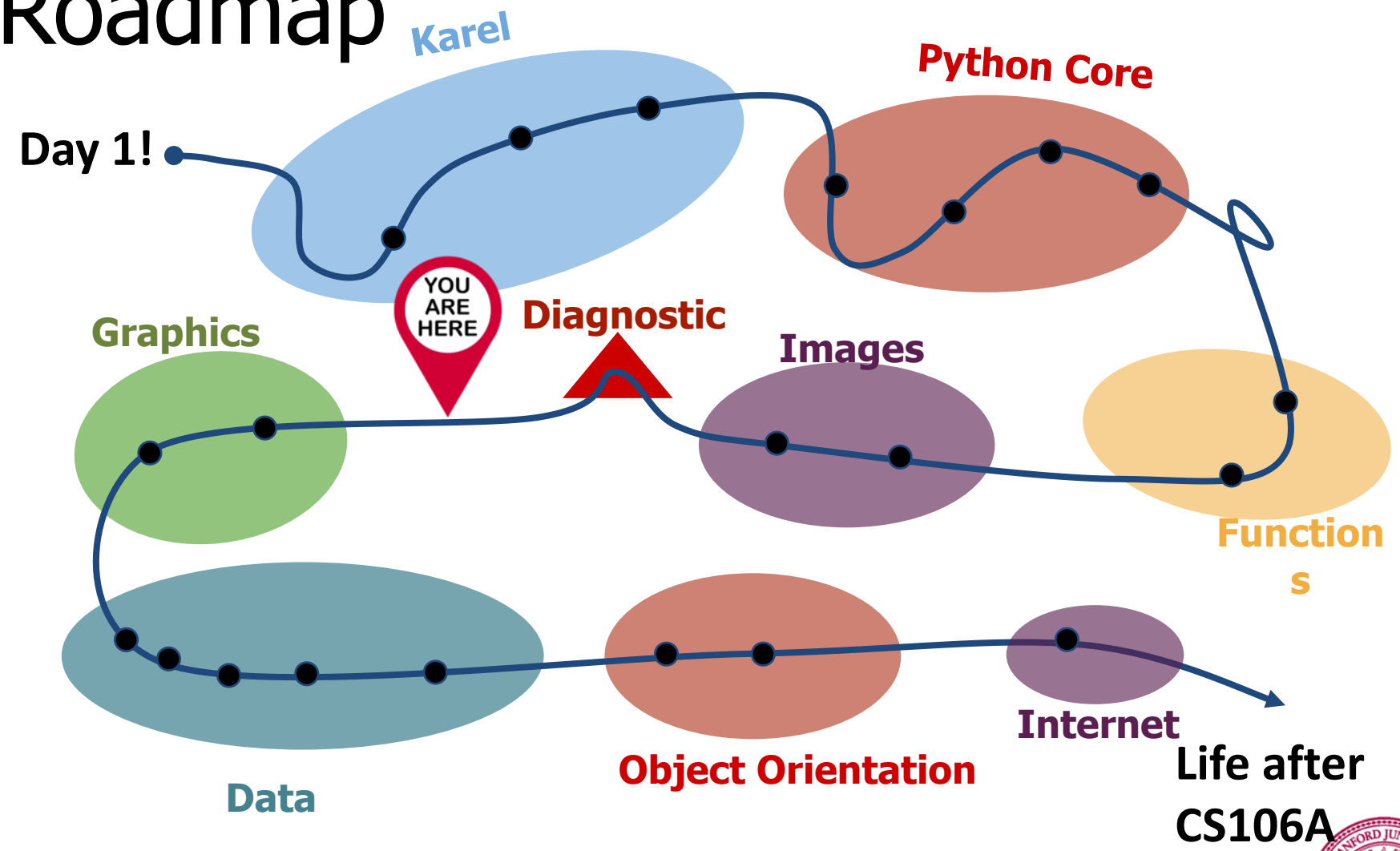
Graphics

Chris Piech and Mehran Sahami
CS106A, Stanford University

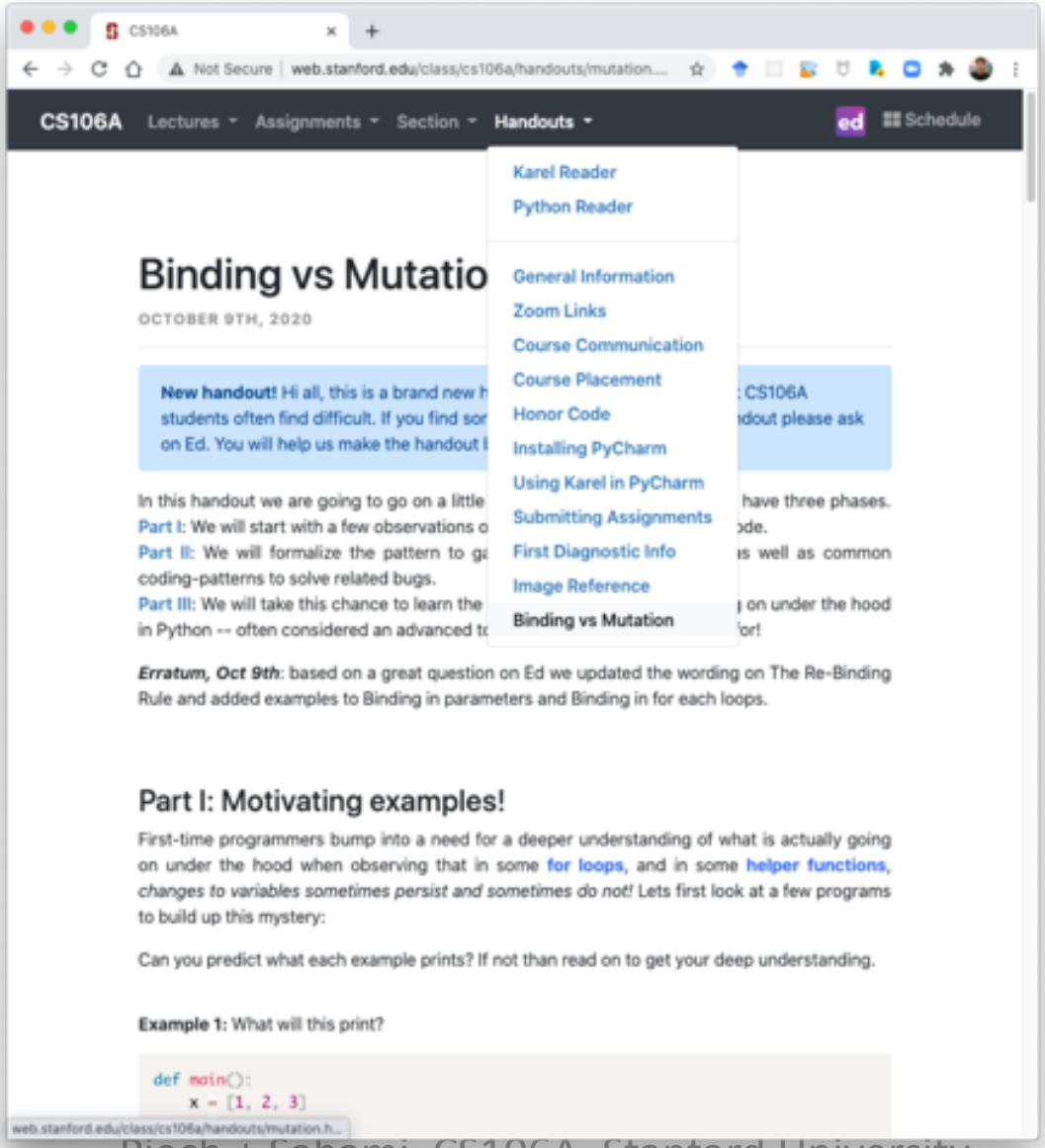
Roadmap


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

Housekeeping

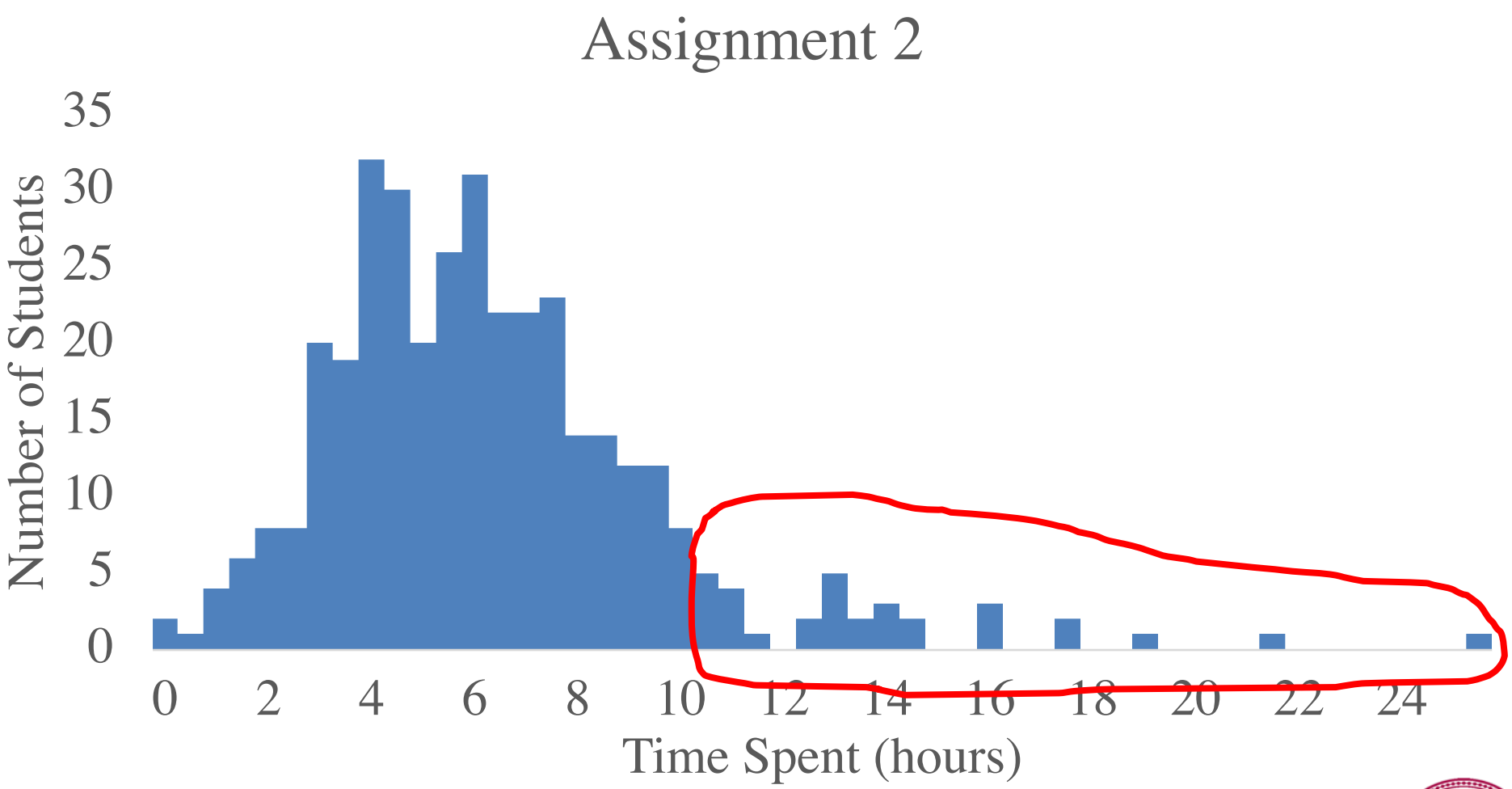
- Give the diagnostic back at the end of lecture
- New handout on Binding vs Mutation

New Handout!

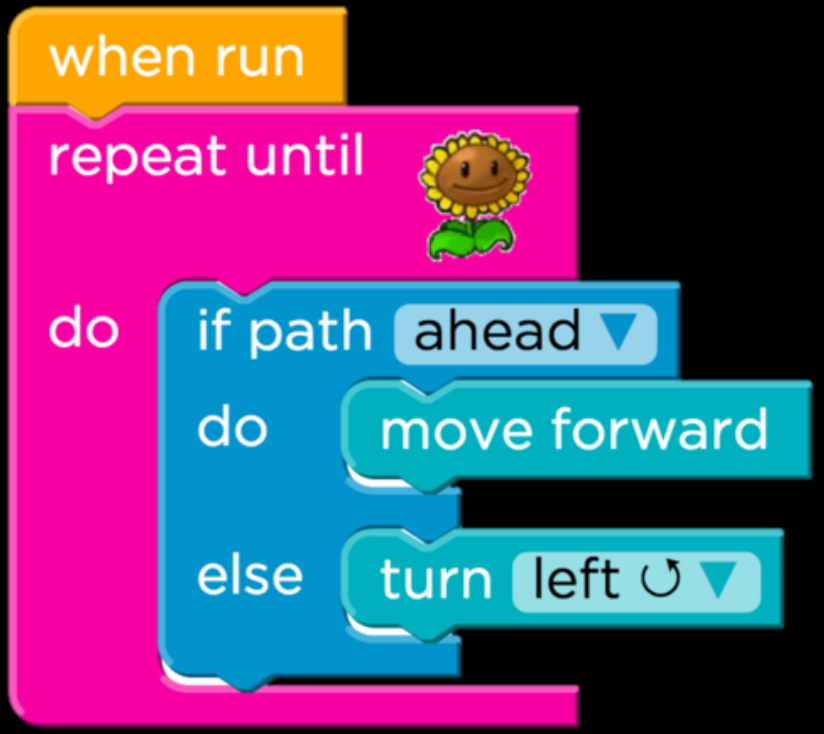

CS106A
Not Secure
web.stanford.edu/class/cs106a/handouts/mutation....
CS106A
Lectures
Assignments
Section
Handouts
ed
Schedule
Karel Reader
Python Reader
General Information
Zoom Links
Course Communication
Course Placement
Honor Code
Installing PyCharm
Using Karel in PyCharm
Submitting Assignments
First Diagnostic Info
Image Reference
Binding vs Mutation
Binding vs Mutatio
OCTOBER 9TH, 2020
New handout! Hi all, this is a brand new h
: CS106A
students often find difficult. If you find sor
dout please ask
on Ed. You will help us make the handout
In this handout we are going to go on a little
have three phases.
Part I: We will start with a few observations o
ode.
Part II: We will formalize the pattern to ga
s well as common
coding-patterns to solve related bugs.
Part III: We will take this chance to learn the
on under the hood
in Python -- often considered an advanced t
or!
Erratum, Oct 9th: based on a great question on Ed we updated the wording on The Re-Binding Rule and added examples to Binding in parameters and Binding in for each loops.
Part I: Motivating examples!
First-time programmers bump into a need for a deeper understanding of what is actually going on under the hood when observing that in some for loops, and in some helper functions, changes to variables sometimes persist and sometimes do not! Lets first look at a few programs to build up this mystery:
Can you predict what each example prints? If not than read on to get your deep understanding.
Example 1: What will this print?
def main():
x = [1, 2, 3]
web.stanford.edu/class/cs106a/handouts/mutation.h...

Get a little meta…

Learn by Doing



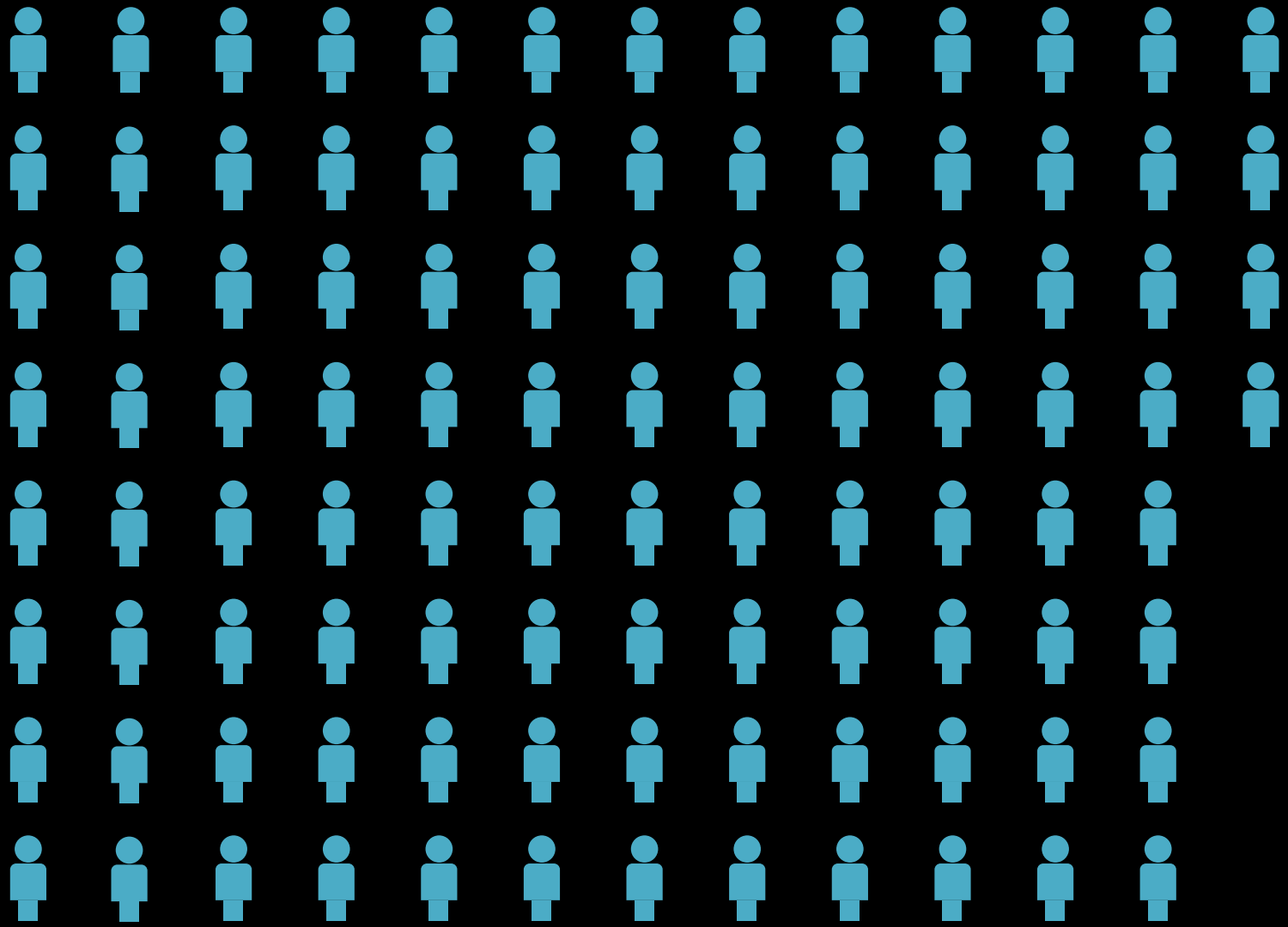
Learning to Program on the Internet



Task

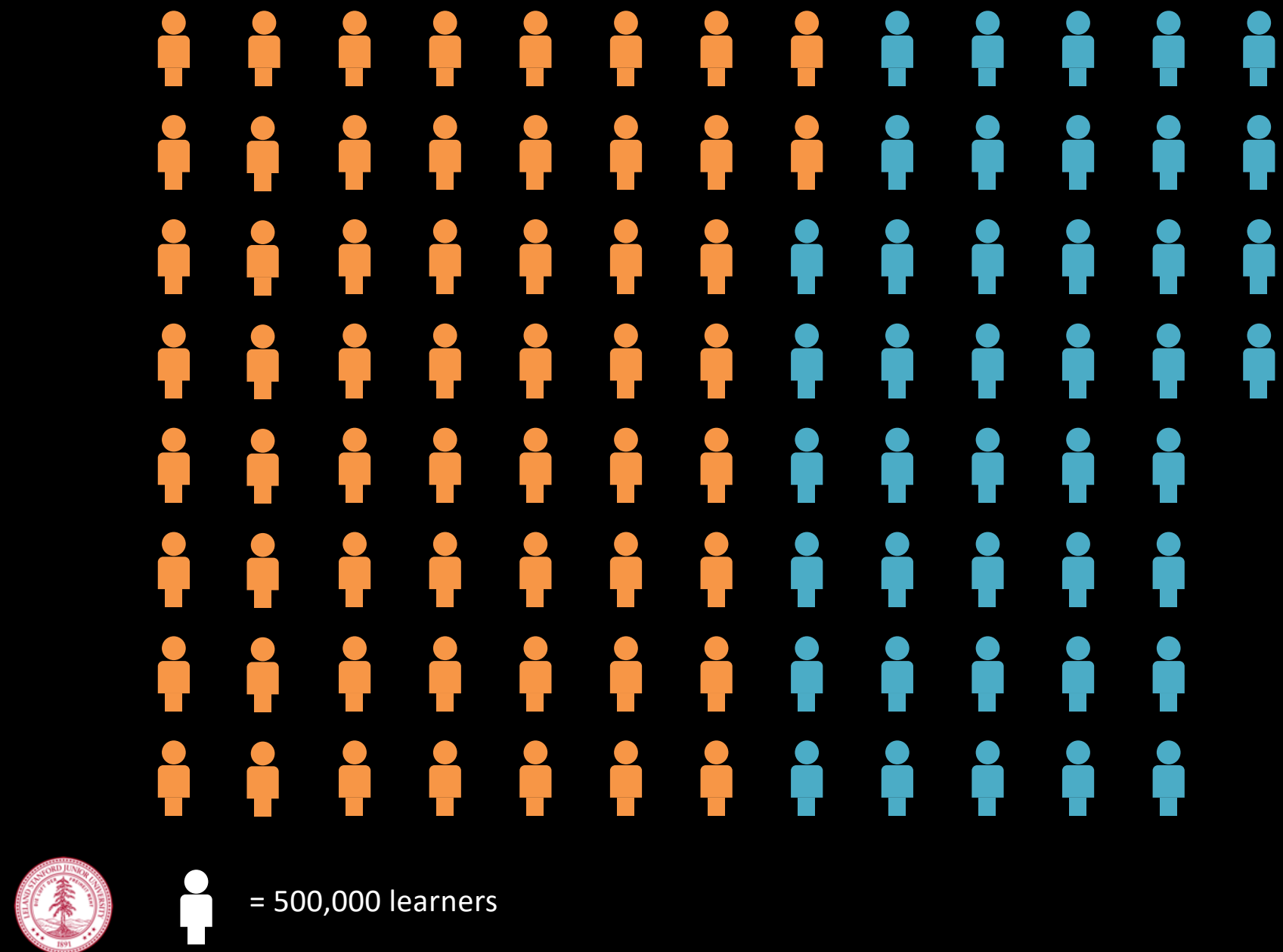
Almost a hundred thousand unique solutions

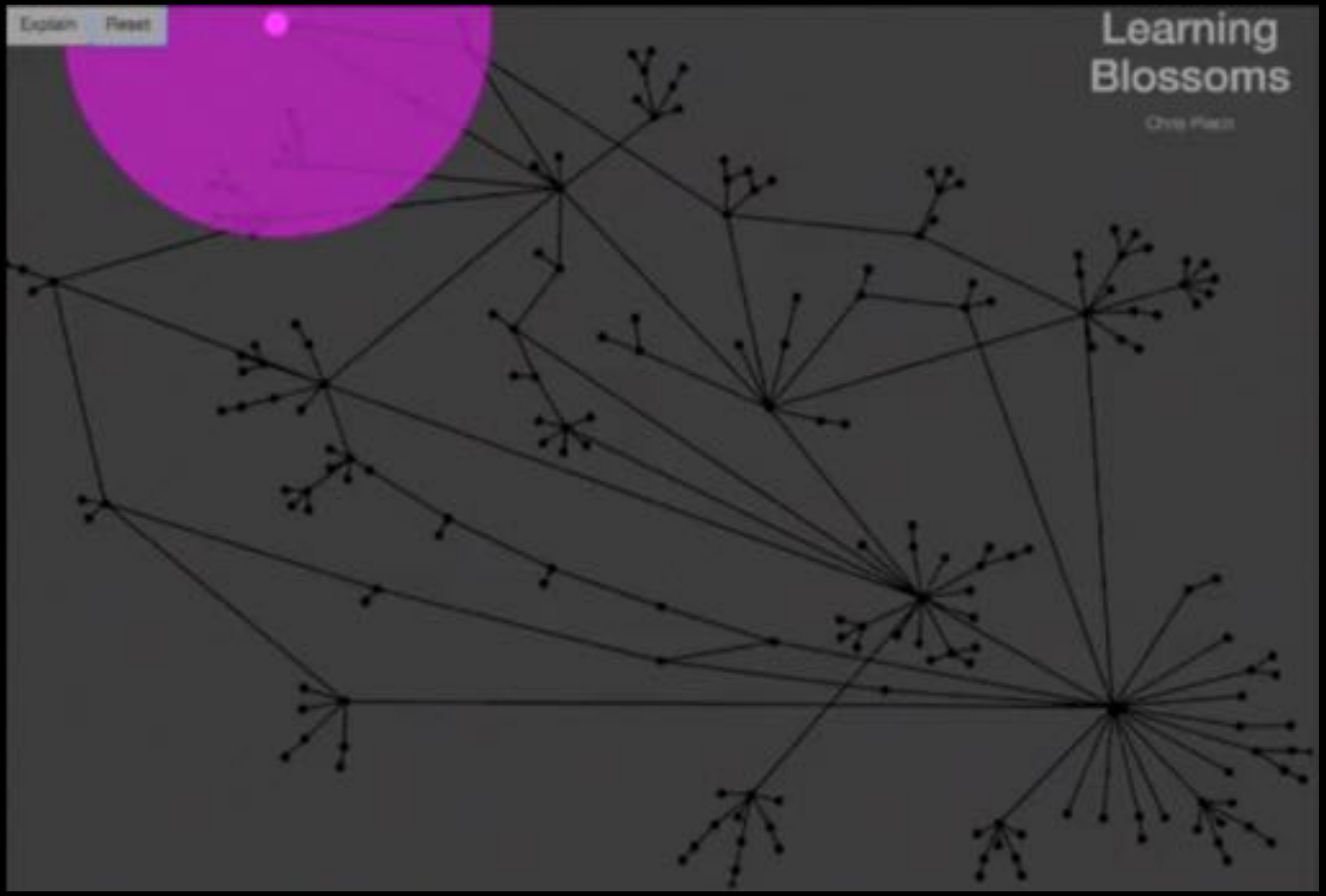
US K-12 Students



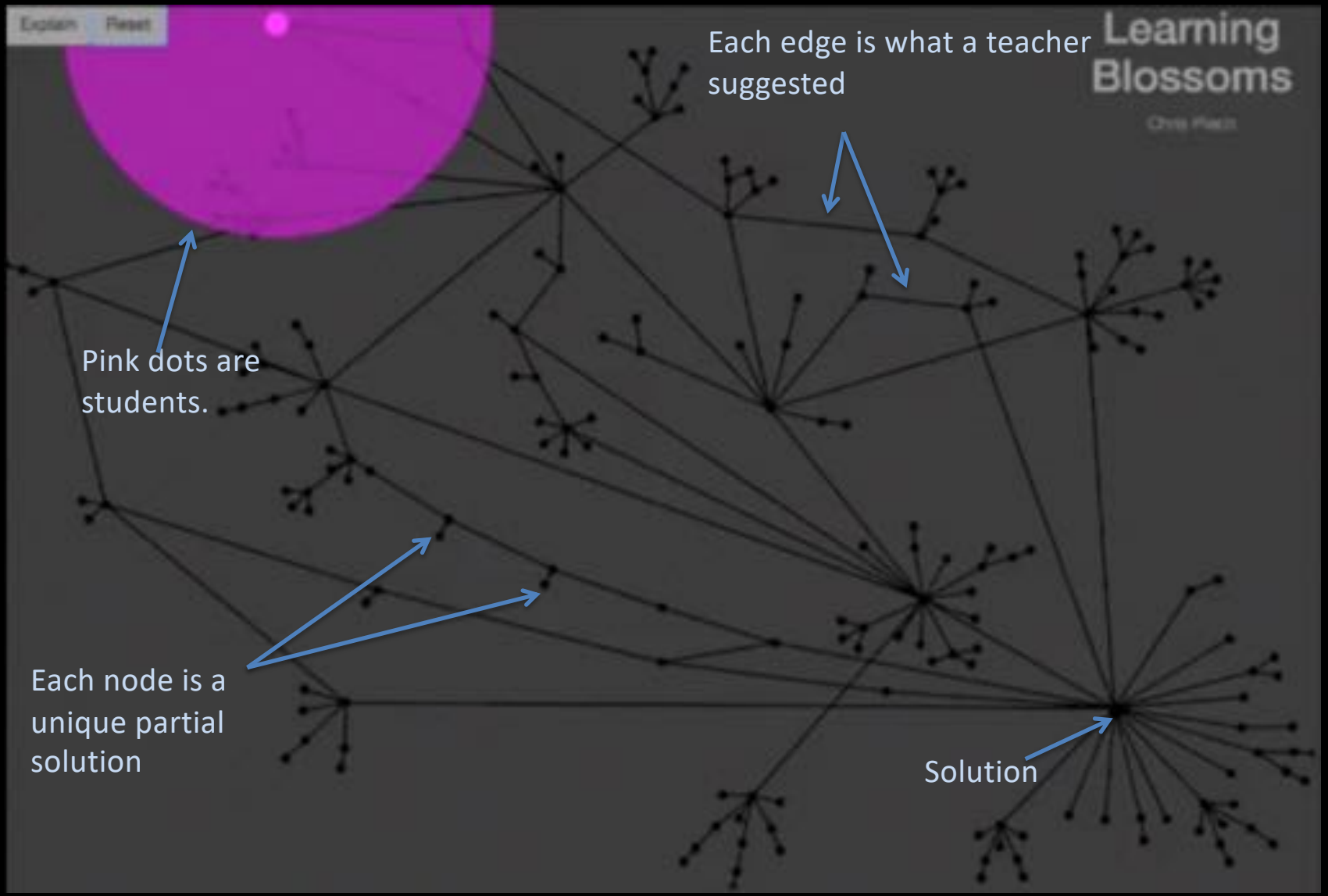
= 500,000 learners

Code.org Students

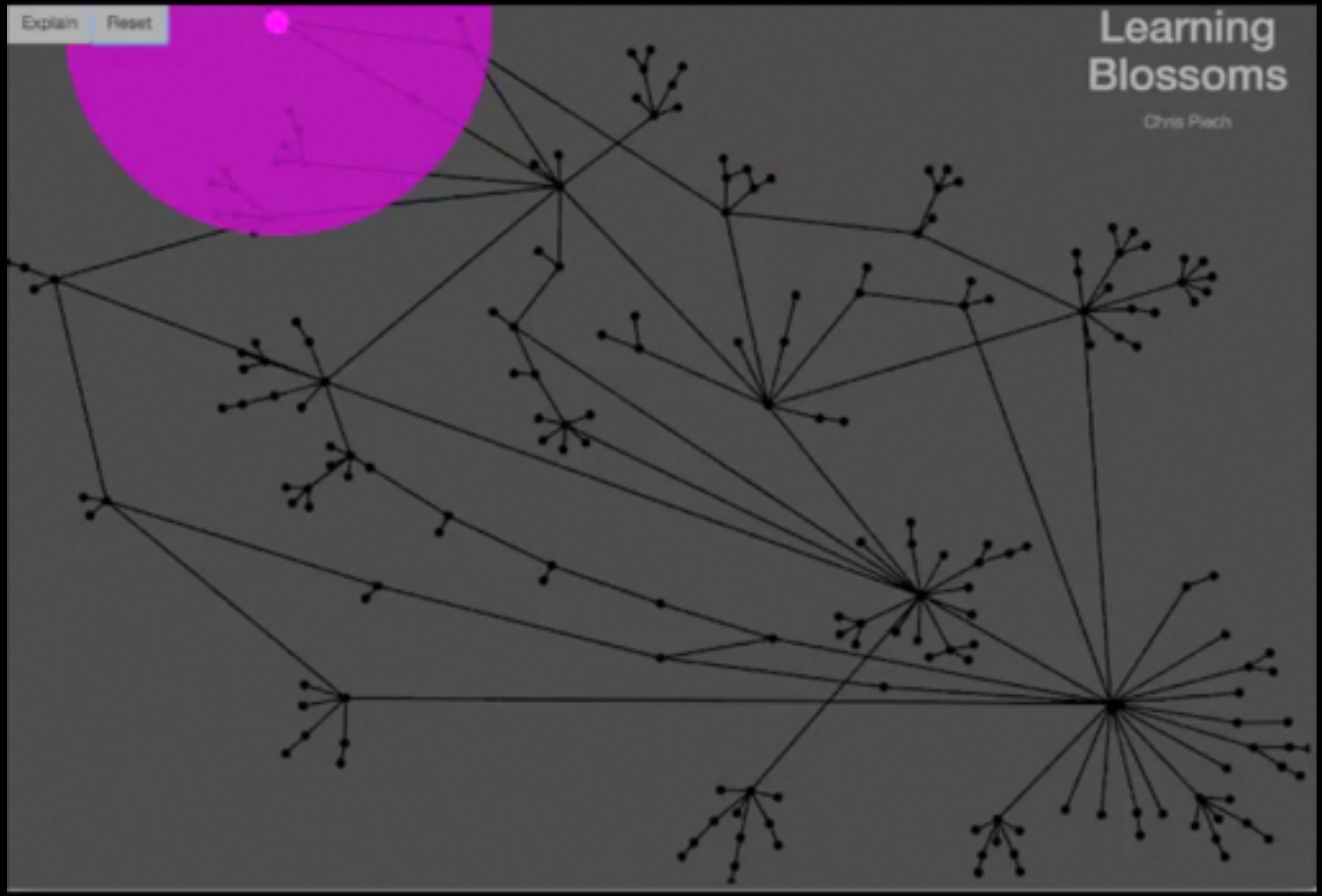




Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



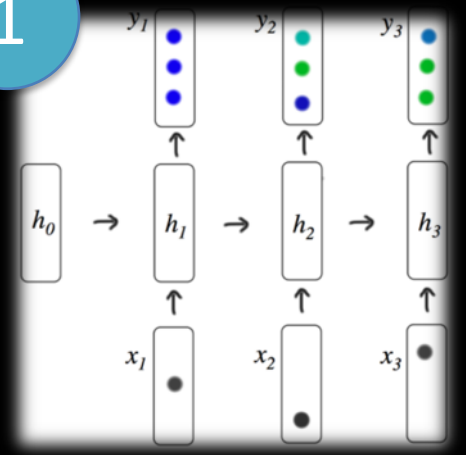
Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



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

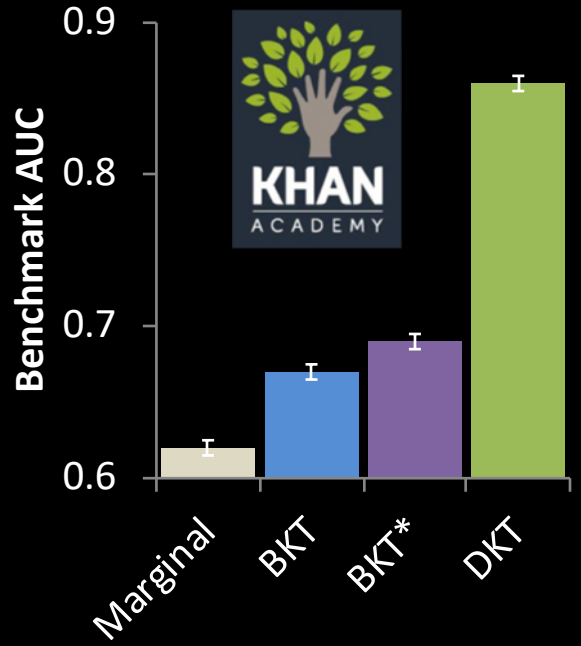
Deep Learning Algorithms

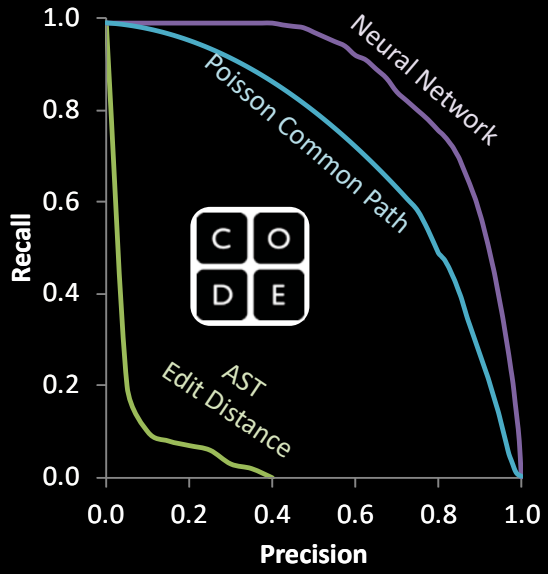
1



2

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

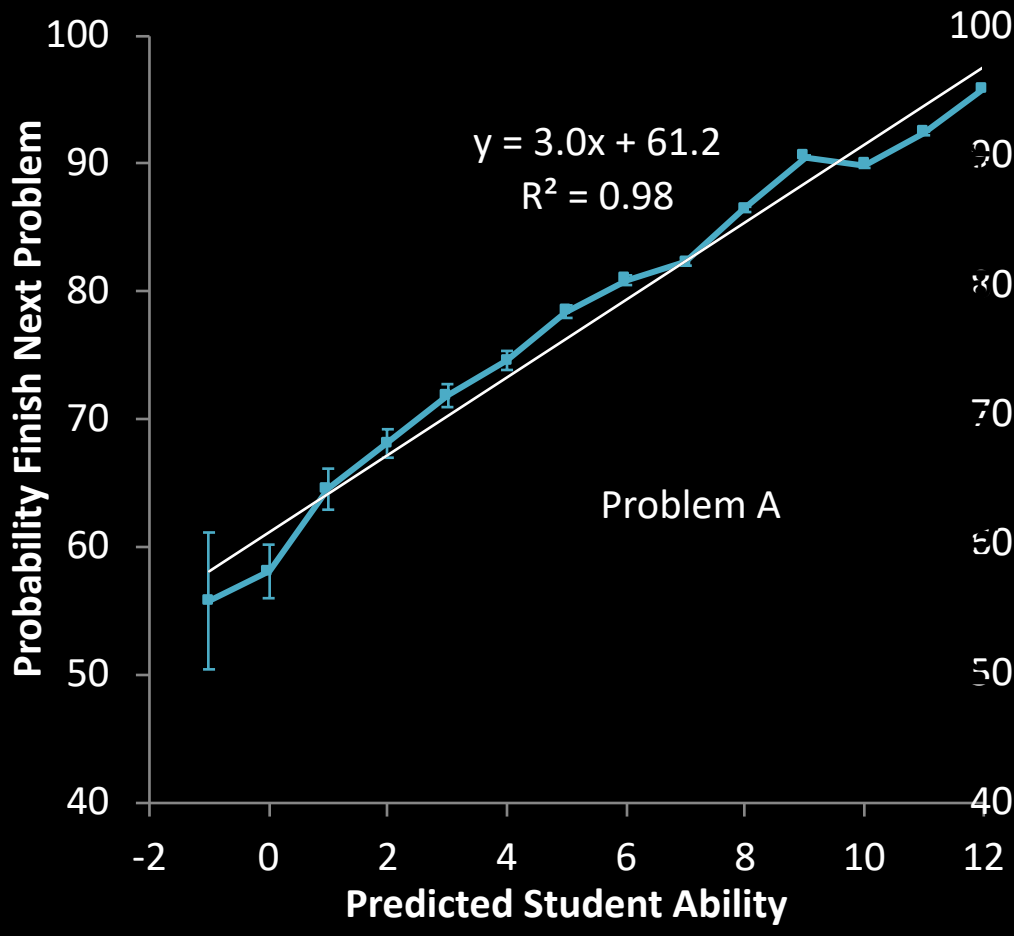


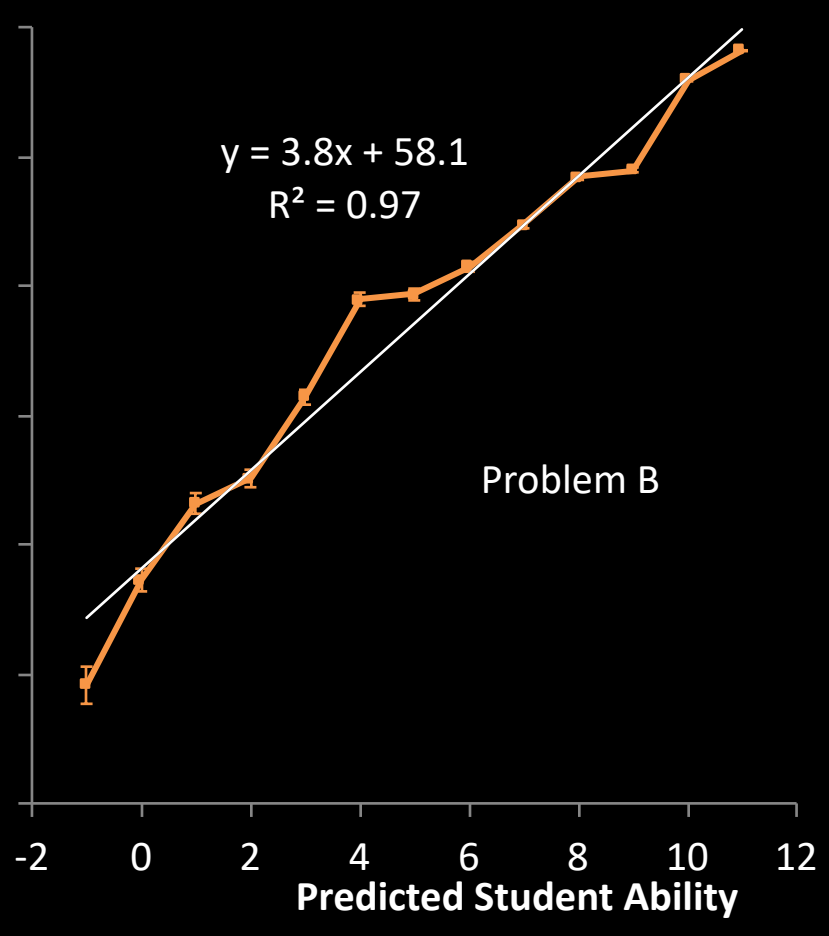


Deep Learning on Trajectories

Research in collaboration with Lisa Wang

Predicts Future Success

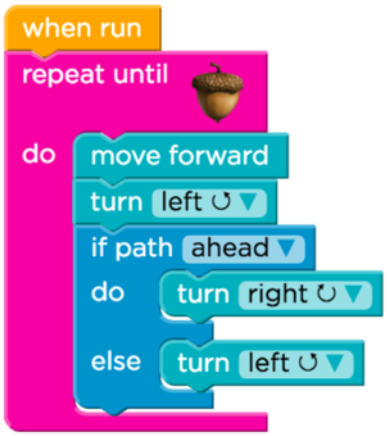




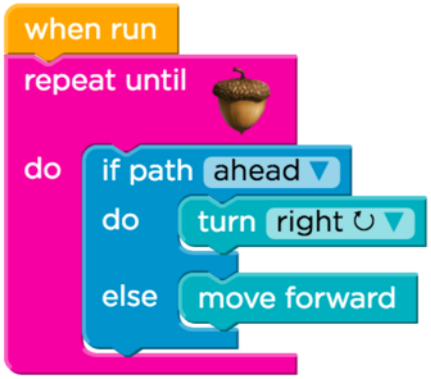
Most likely to succeed?

Highly Rates Grit

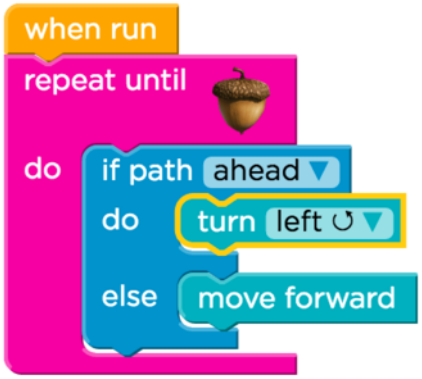
1. Two compound errors



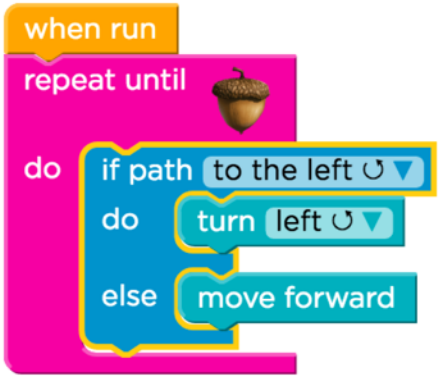
2. Solves first error



3. Starts reasonable attempt



4. Completes attempt

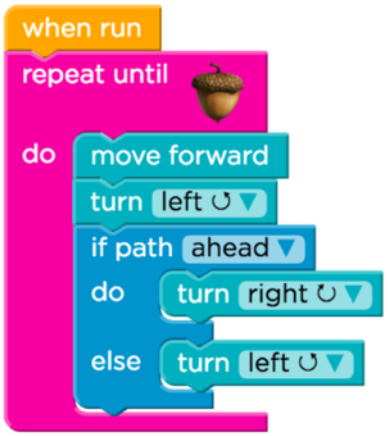


5. Backtracks

6. Finds solution

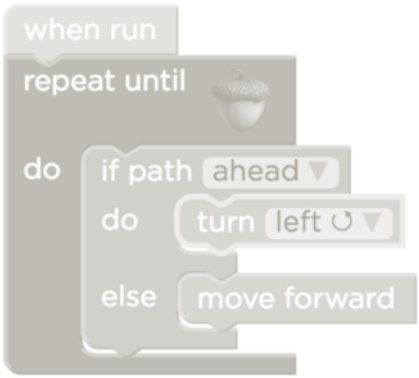
Highly Rates Grit

1. Two compound errors



2. Solves first error

3. Starts reasonable attempt



4. Completes attempt

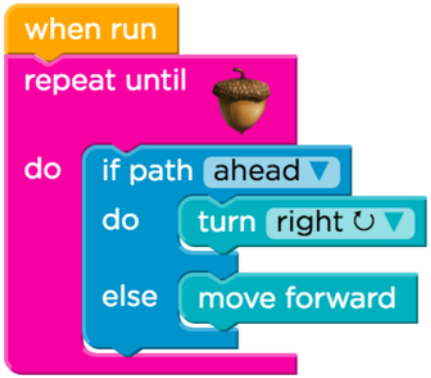
5. Backtracks

6. Finds solution

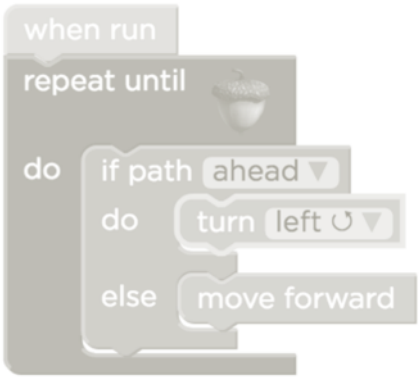
Highly Rates Grit

1. Two compound errors

2. Solves first error



3. Starts reasonable attempt



4. Completes attempt

5. Backtracks

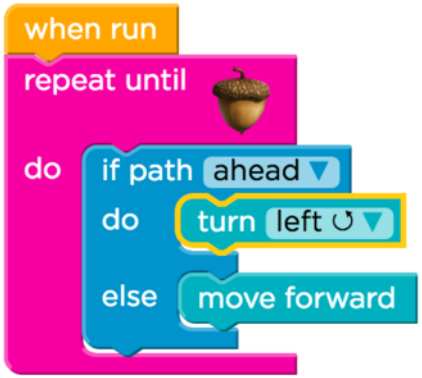
6. Finds solution

Highly Rates Grit

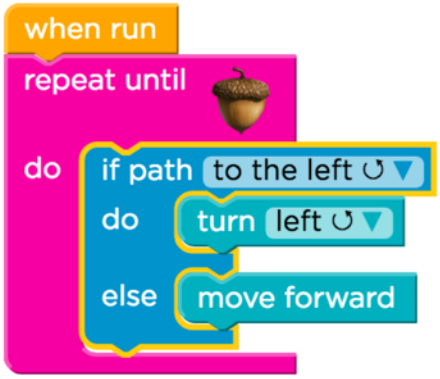
1. Two compound errors

2. Solves first error

3. Starts reasonable attempt



4. Completes attempt



5. Backtracks

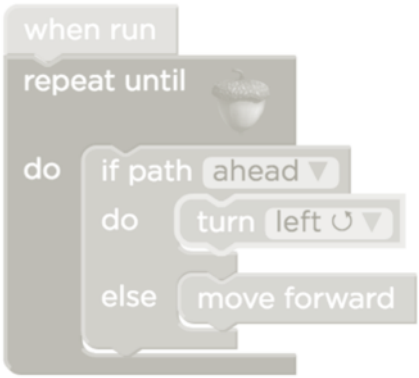
6. Finds solution

Highly Rates Grit

1. Two compound errors

2. Solves first error

3. Starts reasonable attempt



4. Completes attempt

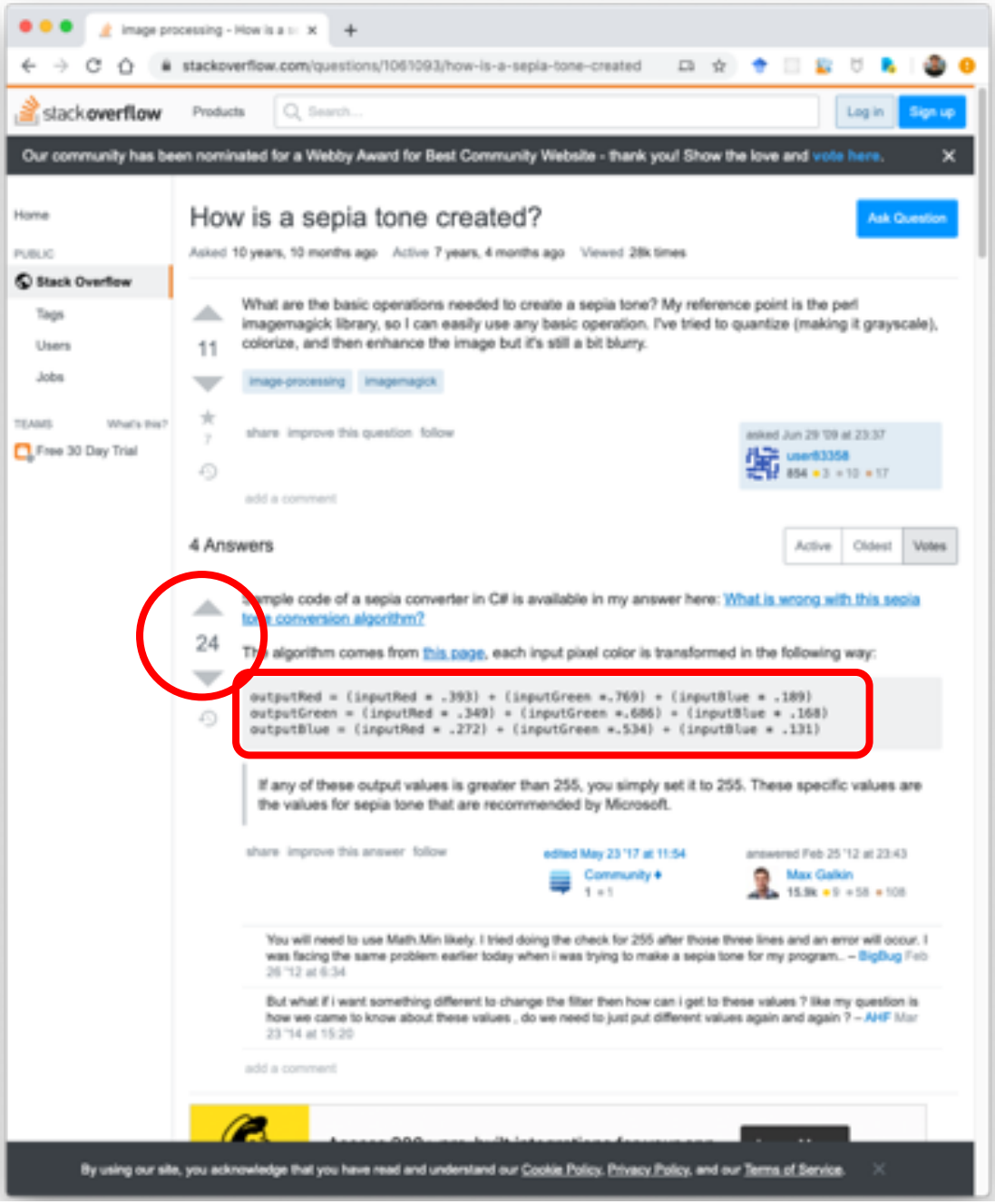
5. Backtracks

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()
```

If you are unsure why this actually changes the image, check out binding vs mutation!

```
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)
```


Sepia Example

```
def main():


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

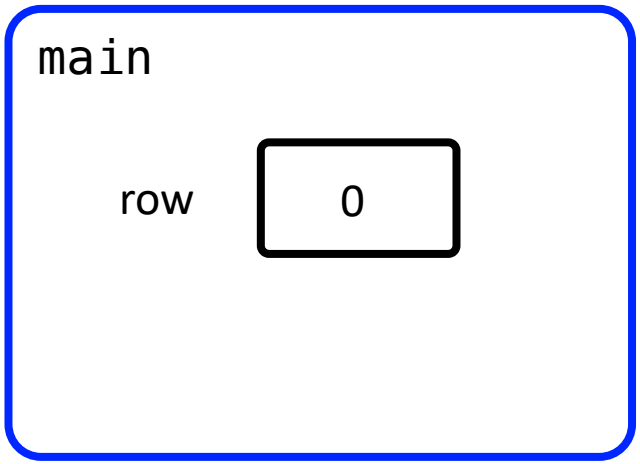
Sepia Example

```
def main():


    for row in range(600):
        for col in range(800):
            print(row, col)
```

Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

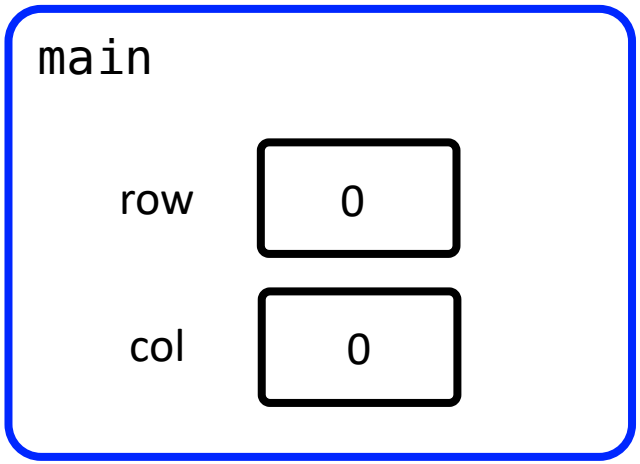
Memory “stack”



Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

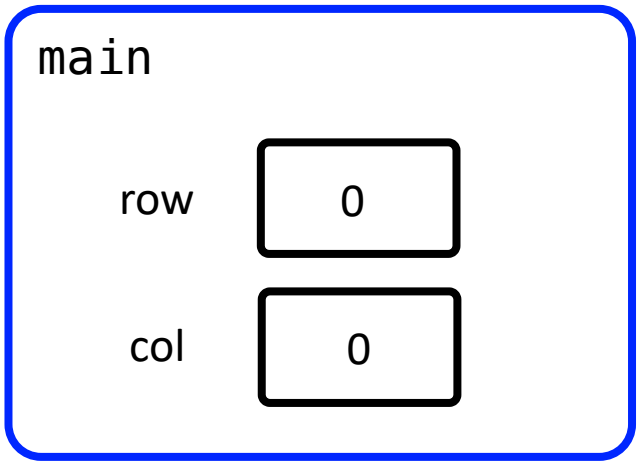
Memory “stack”

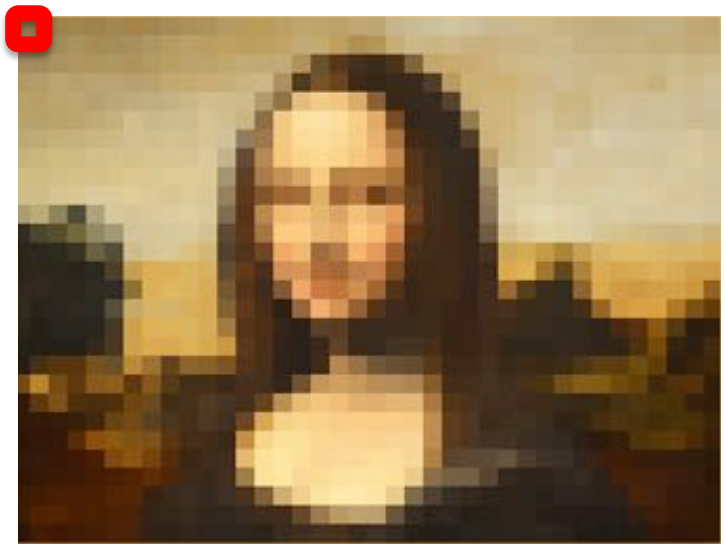


Sepia Example

```
def main():


    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

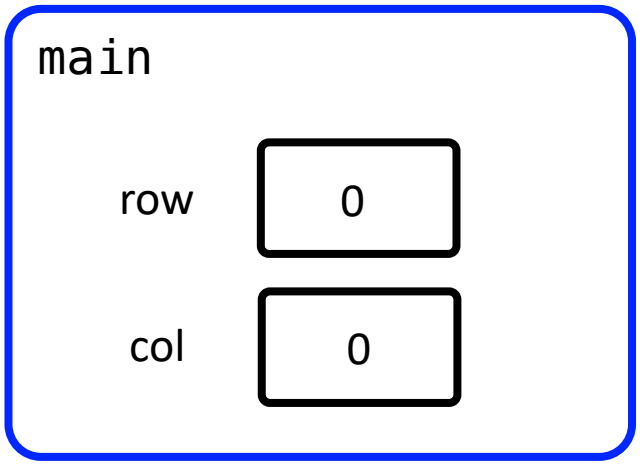


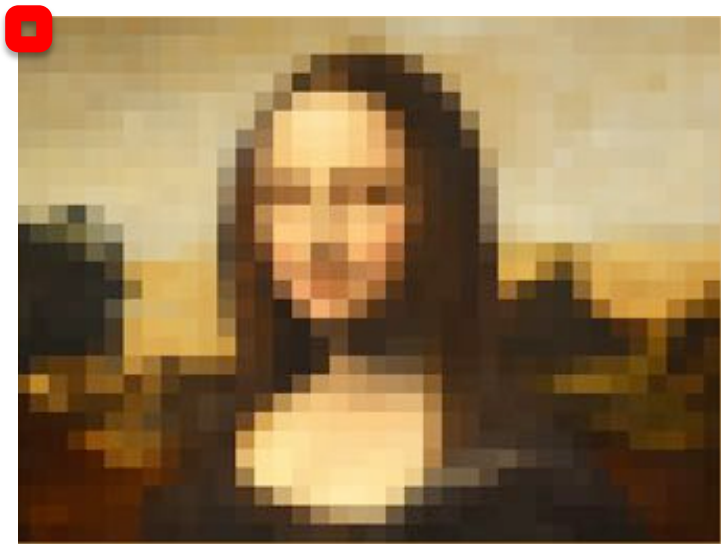


Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

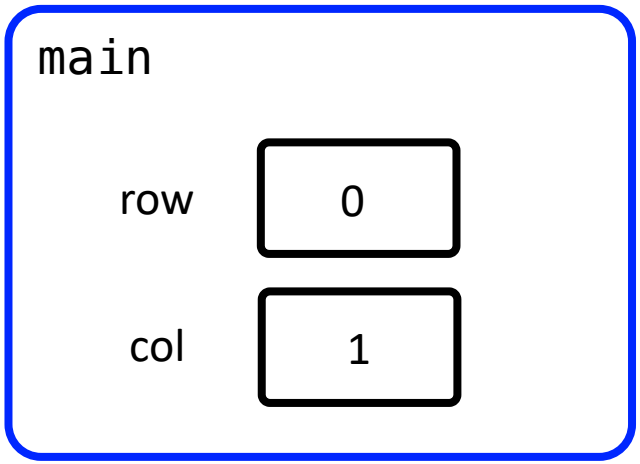


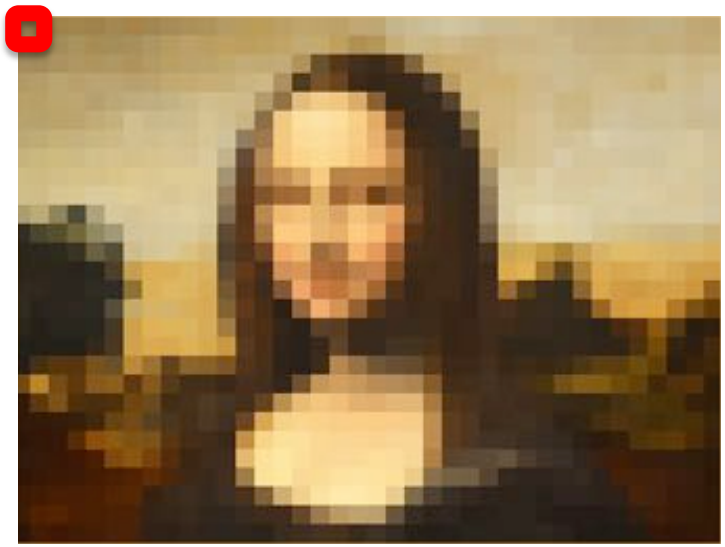


Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

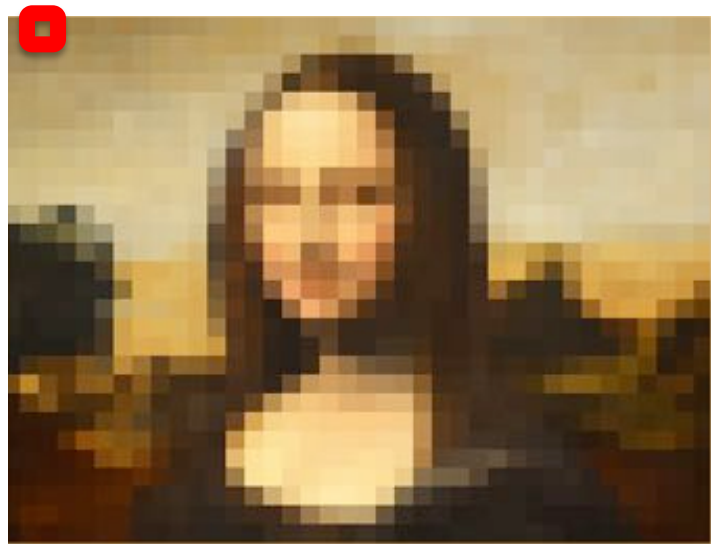




Sepia Example

```
def main():


    for row in range(600):
        for col in range(800):
            print(row, col)
```

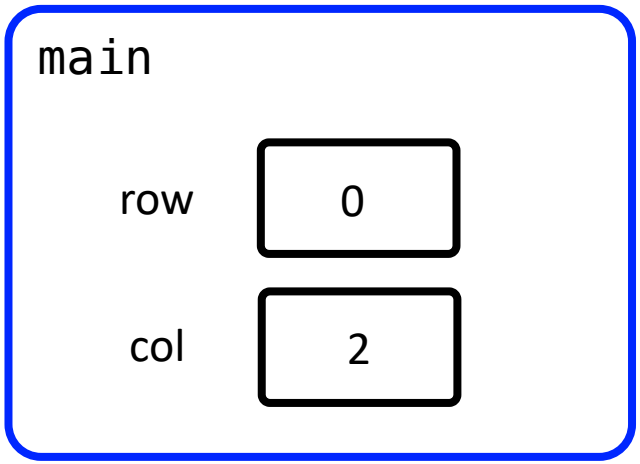
Memory “stack”

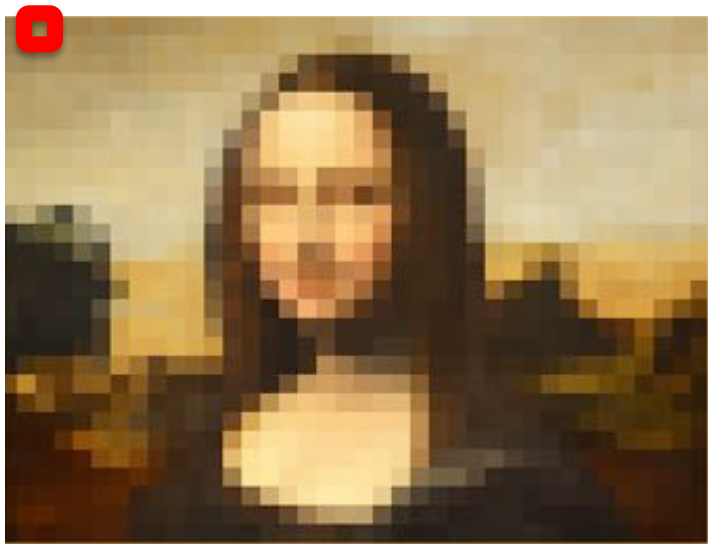


Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

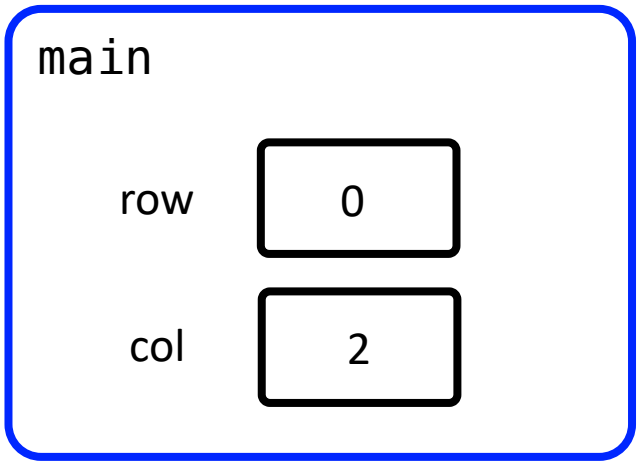


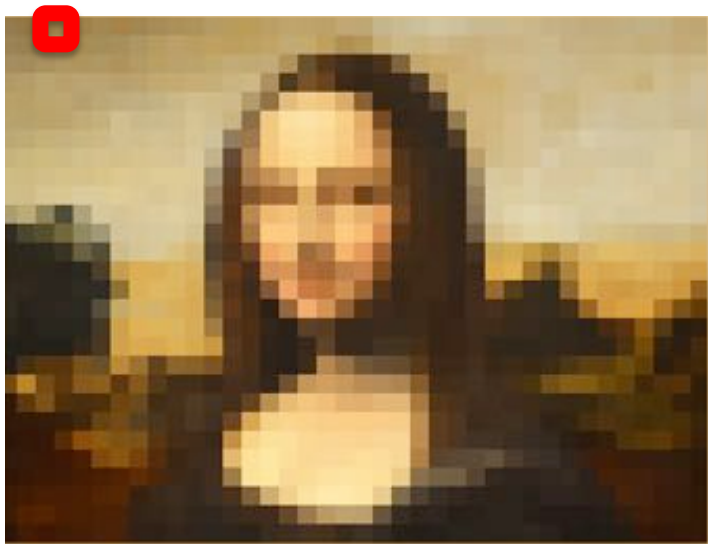


Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

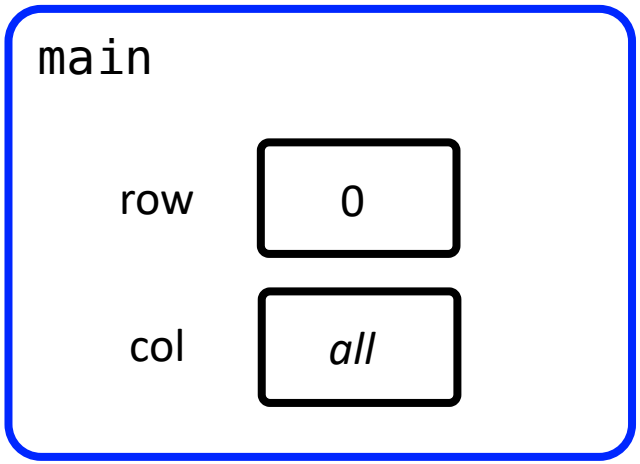




Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

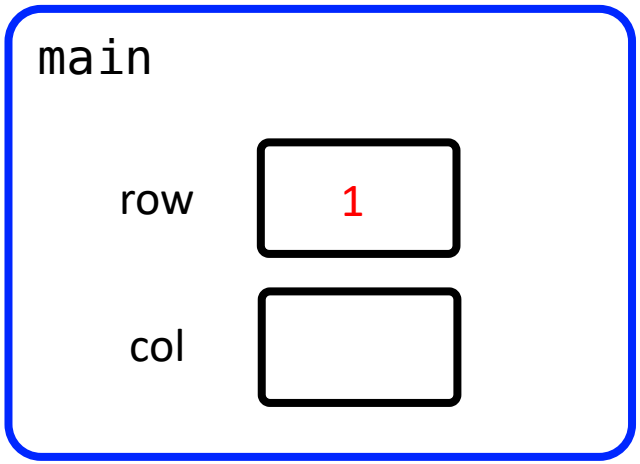
Memory “stack”



Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

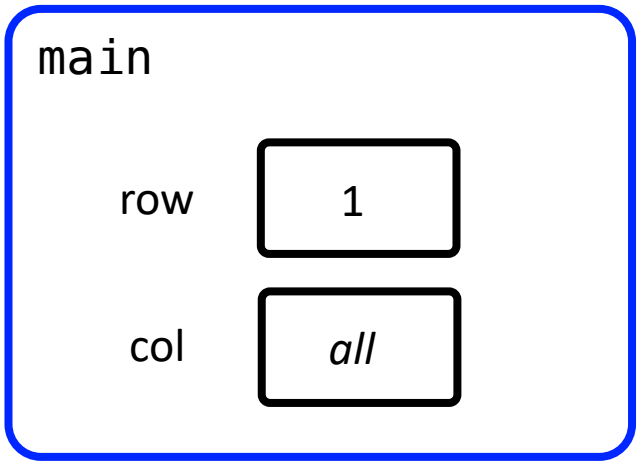
Memory “stack”



Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

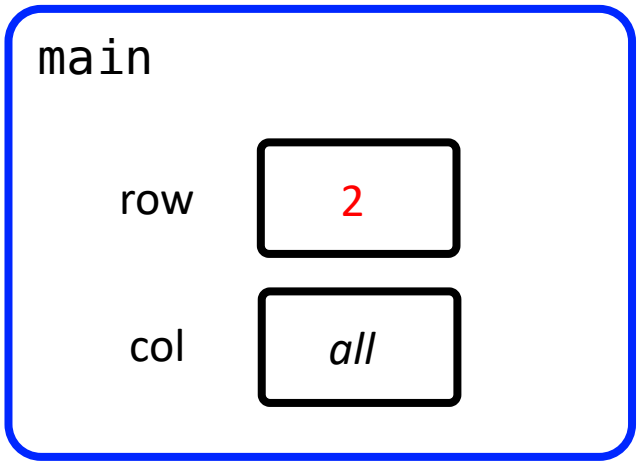
Memory “stack”

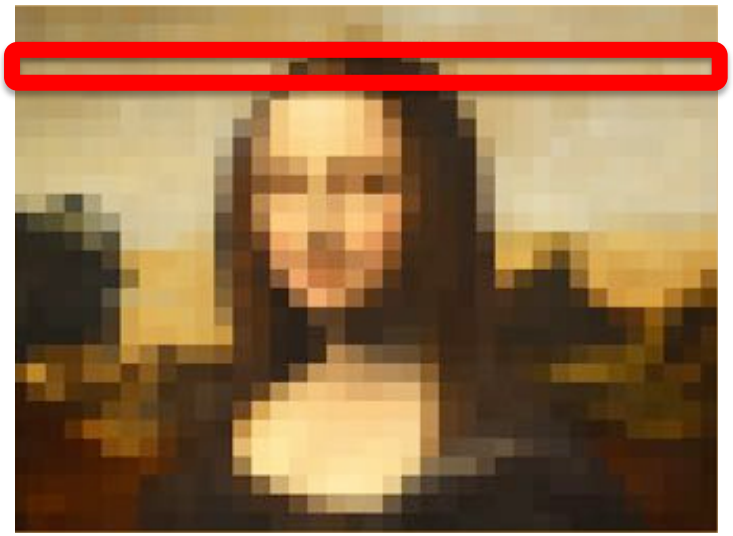


Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”

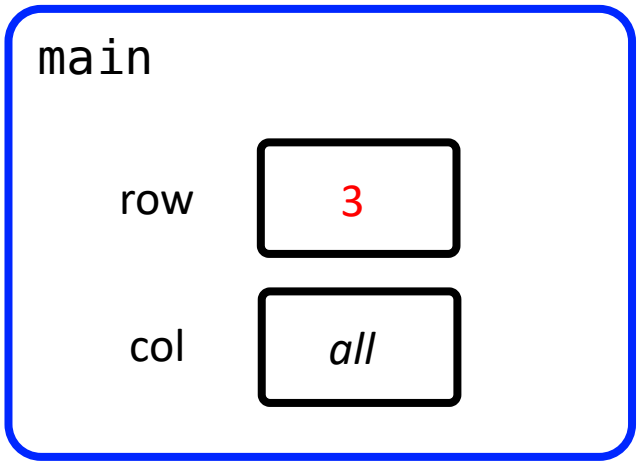




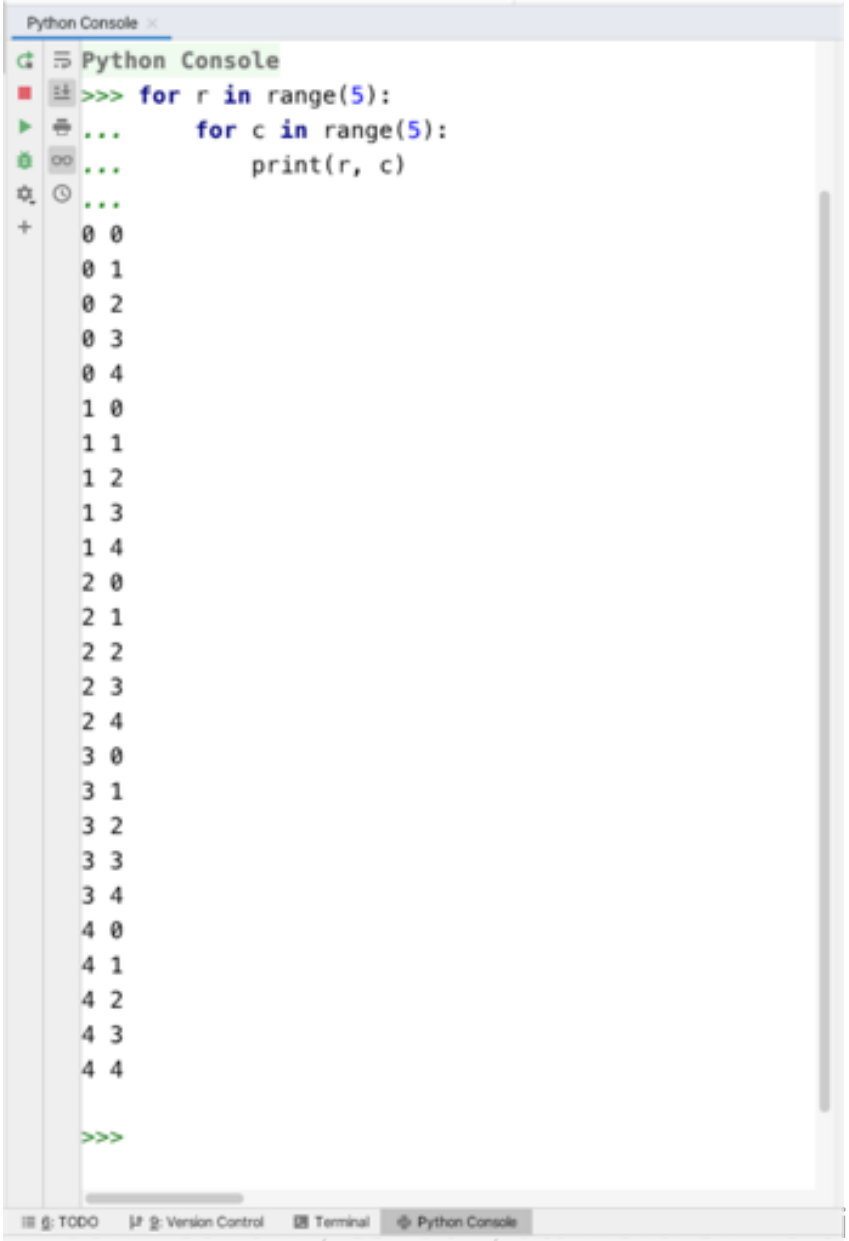
Sepia Example

```
def main():

    for row in range(600):
        for col in range(800):
            print(row, col)
```

Memory “stack”



Python Console



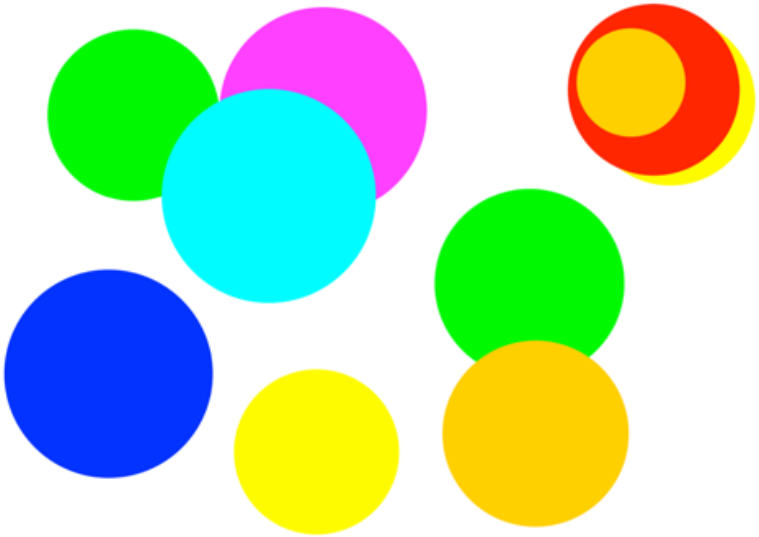
Mike Krieger **Kevin Systrom**

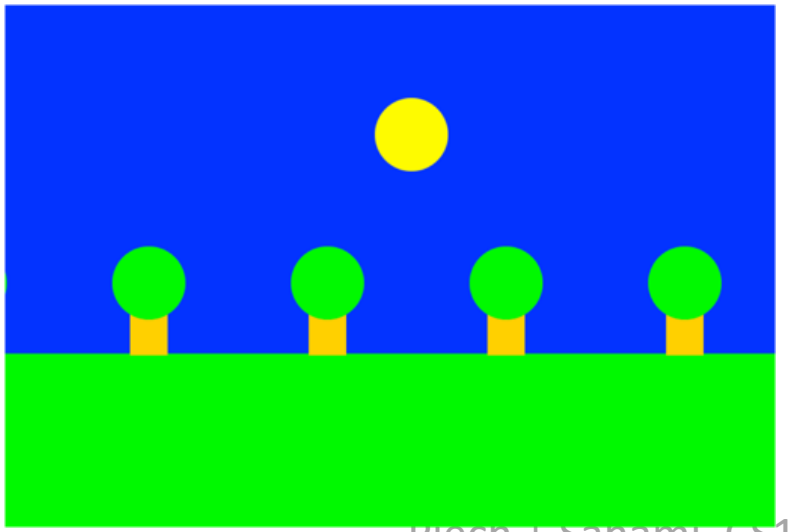
End Review

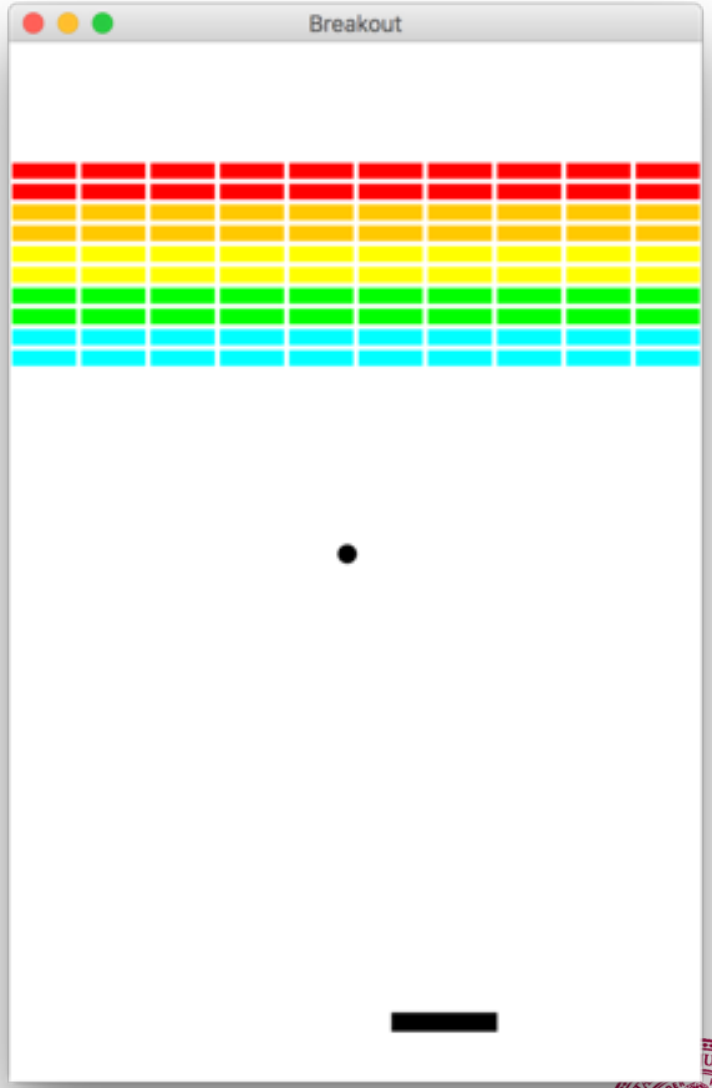
Today's Goal

1. How do I draw shapes?

Graphics Programs



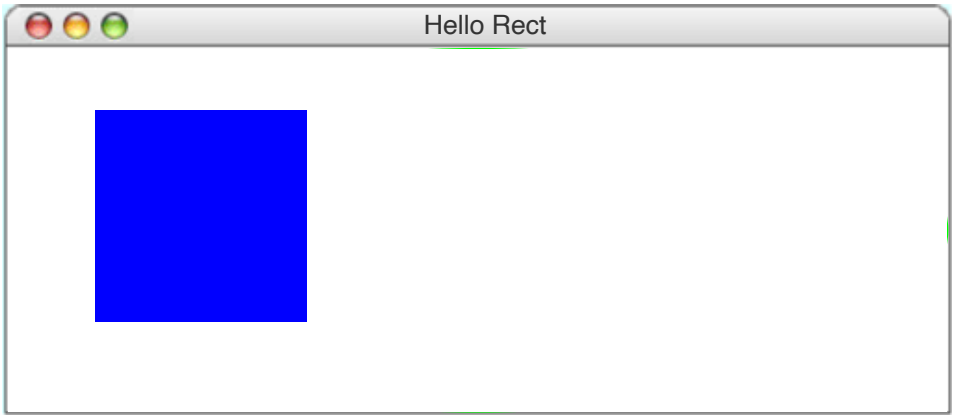




Draw a Rectangle

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

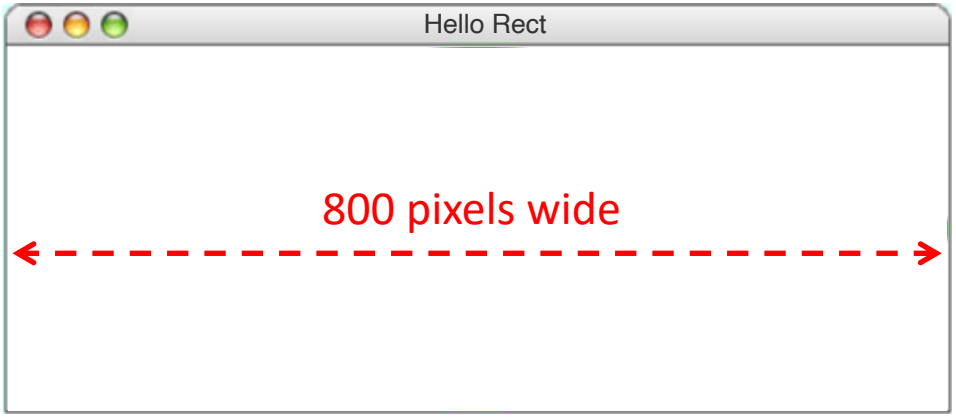
the following **`main`** method displays a blue square

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

Draw a Rectangle

the following `main` method displays a blue square

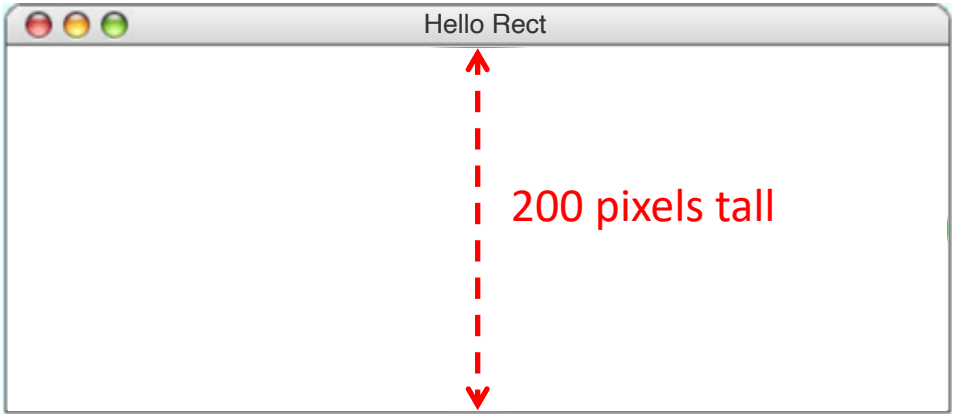
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
```



Draw a Rectangle

the following `main` method displays a blue square

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



Draw a Rectangle

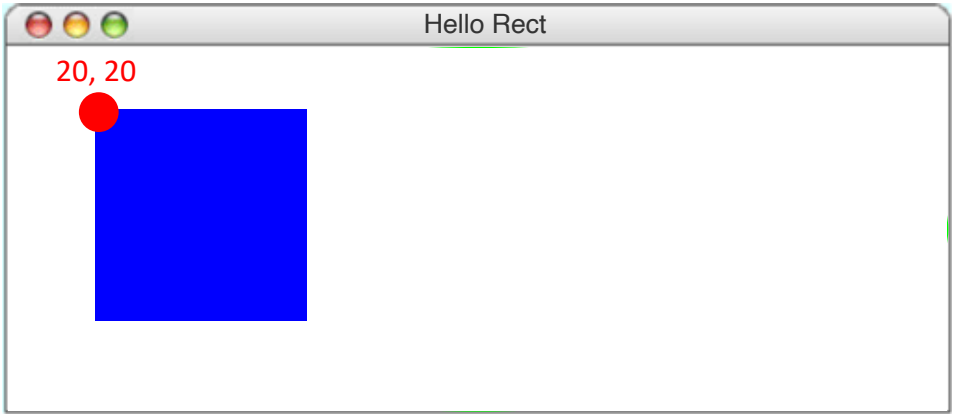
the following **`main`** method displays a blue square

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

Draw a Rectangle

the following `main` method displays a blue square

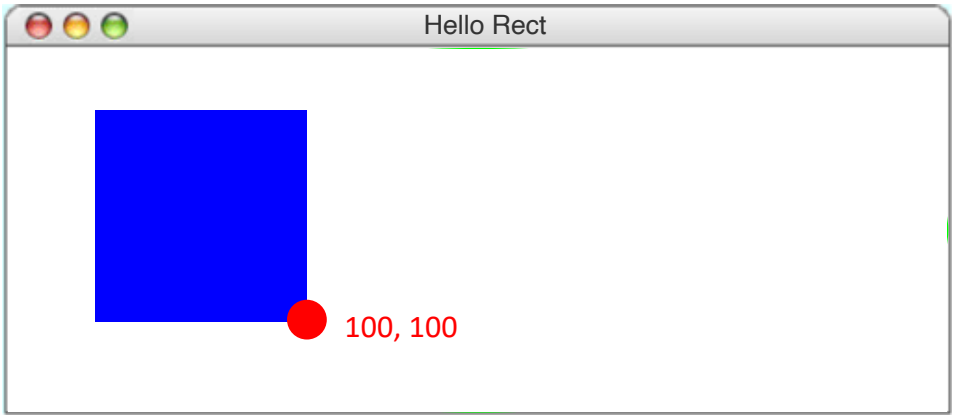
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



Draw a Rectangle

the following `main` method displays a blue square

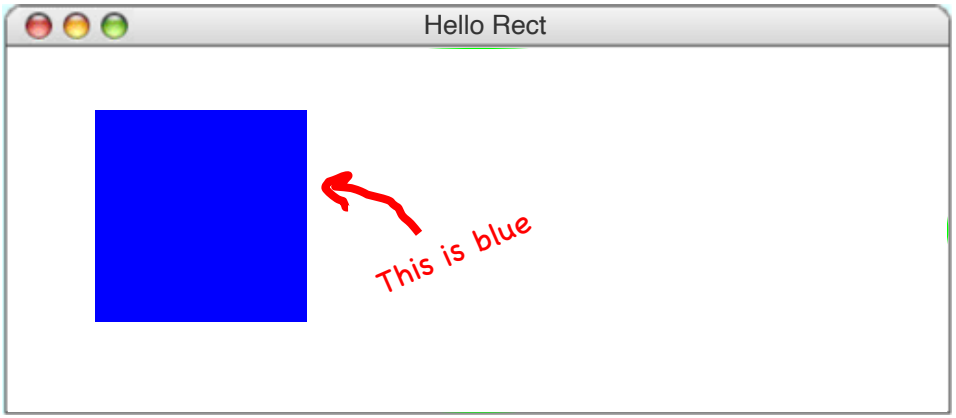
```
def main():
    canvas = make_canvas(800, 200, 'Hello Rect')
    canvas.create_rectangle(20, 20, 100, 100, fill="blue")
```



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")
```

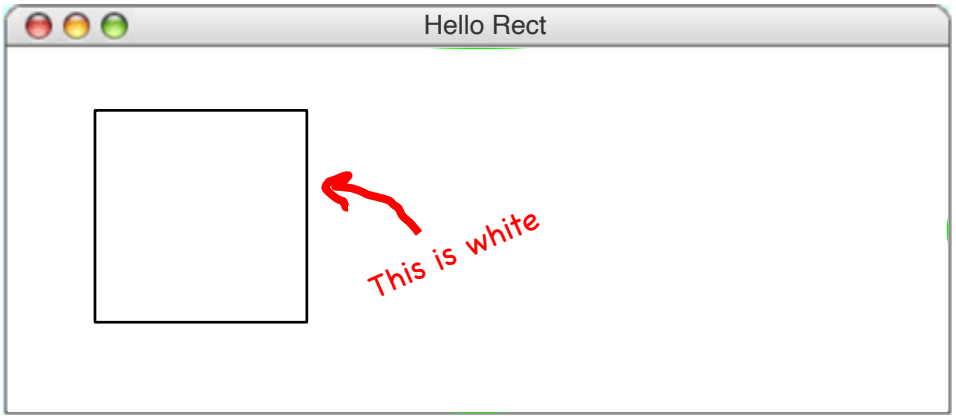


Aside: Named Arguments
This argument is named as `fill`*. It allows functions to have arguments which you can ignore if you want a default value.*

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)
```

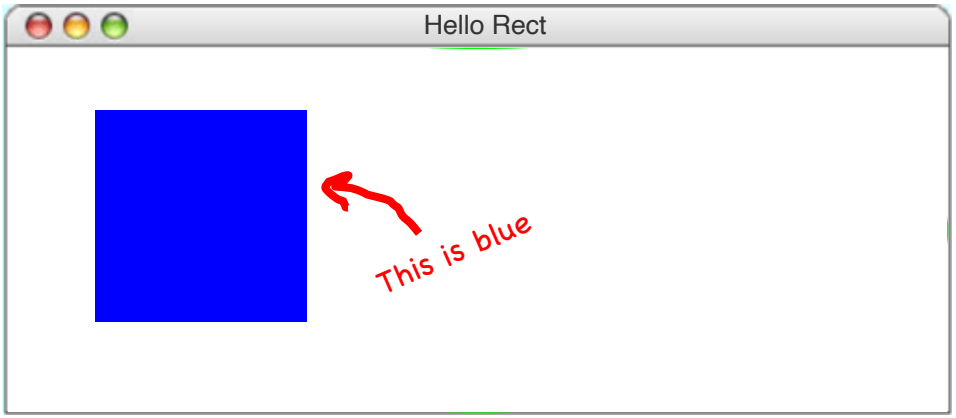


Aside: Named Arguments
This argument is named as `fill`*. It allows functions to have arguments which you can ignore if you want a default value.*

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")
```

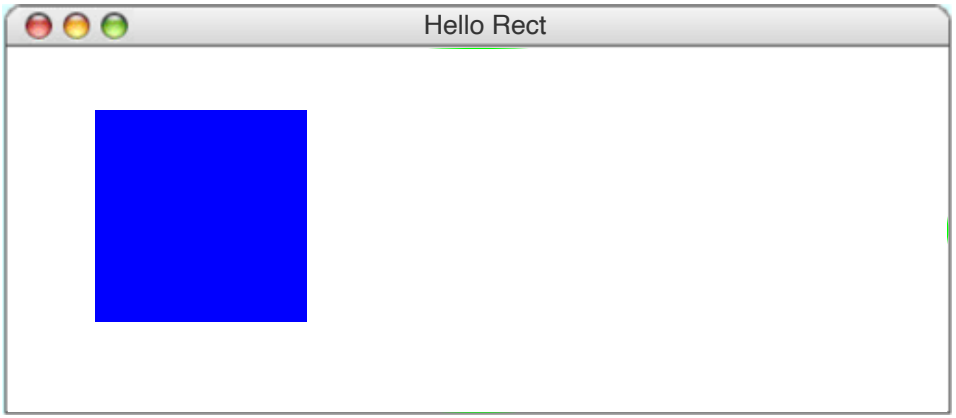


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

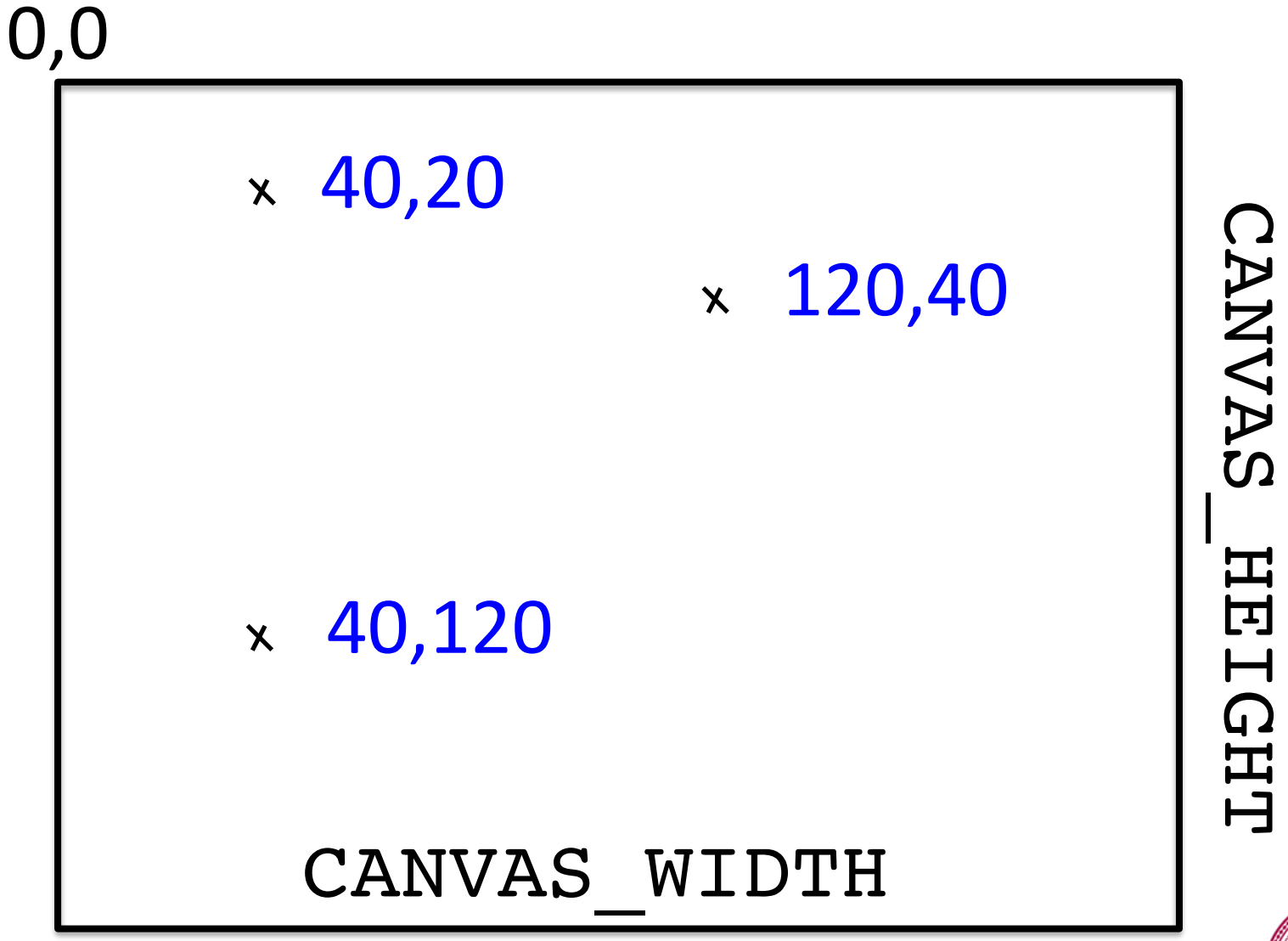
the following **main** method displays a blue square

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

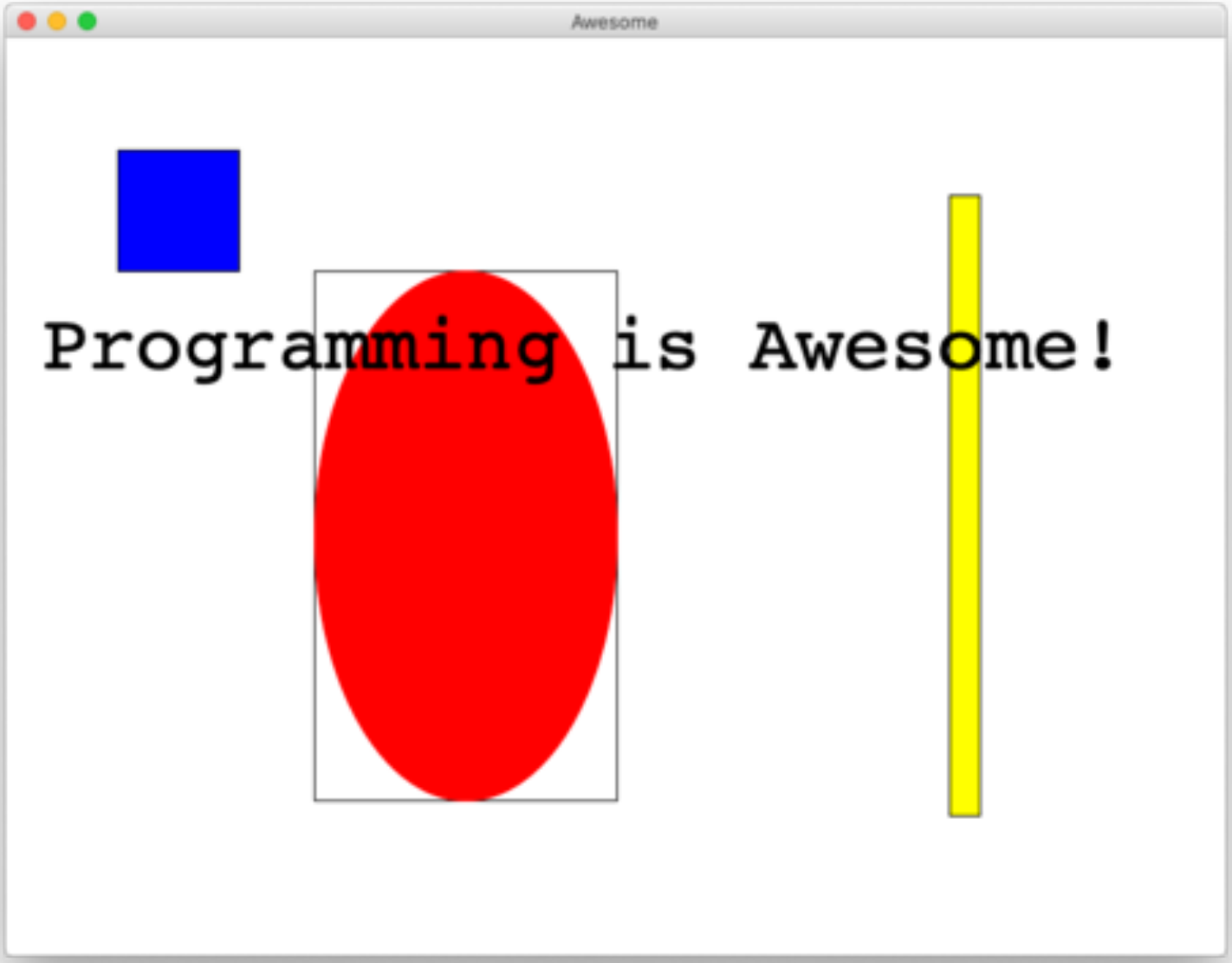


TK Natural Graphics

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)`

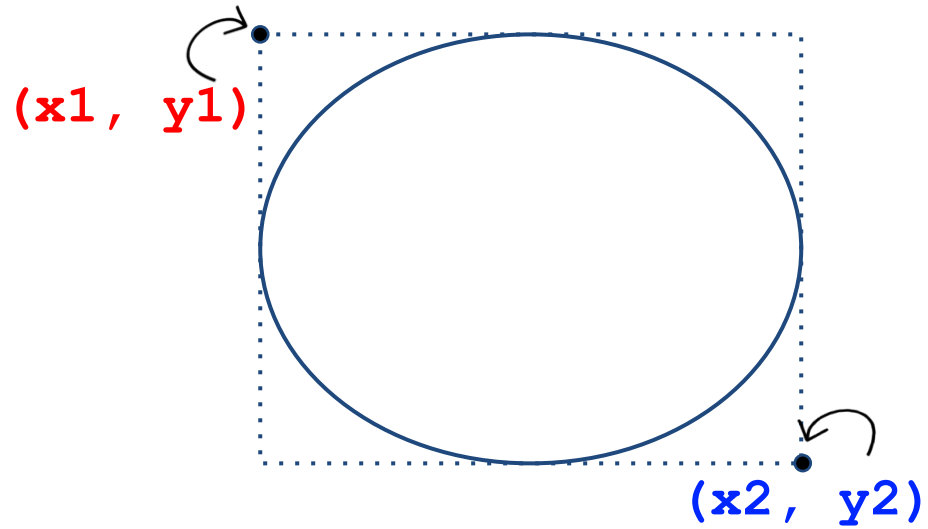
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

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

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

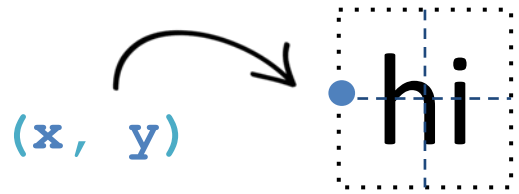
- `canvas.create_line()`
- **`canvas.create_oval(x1, y1, x2, y2)`**
- `canvas.create_text()`

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



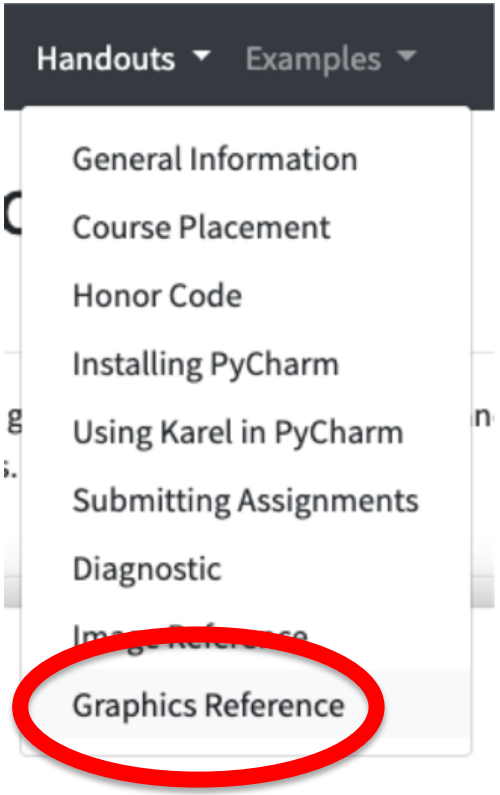
- `canvas.create_line()`
- `canvas.create_oval()`
- **`canvas.create_text(x, y, text='hi')`**

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

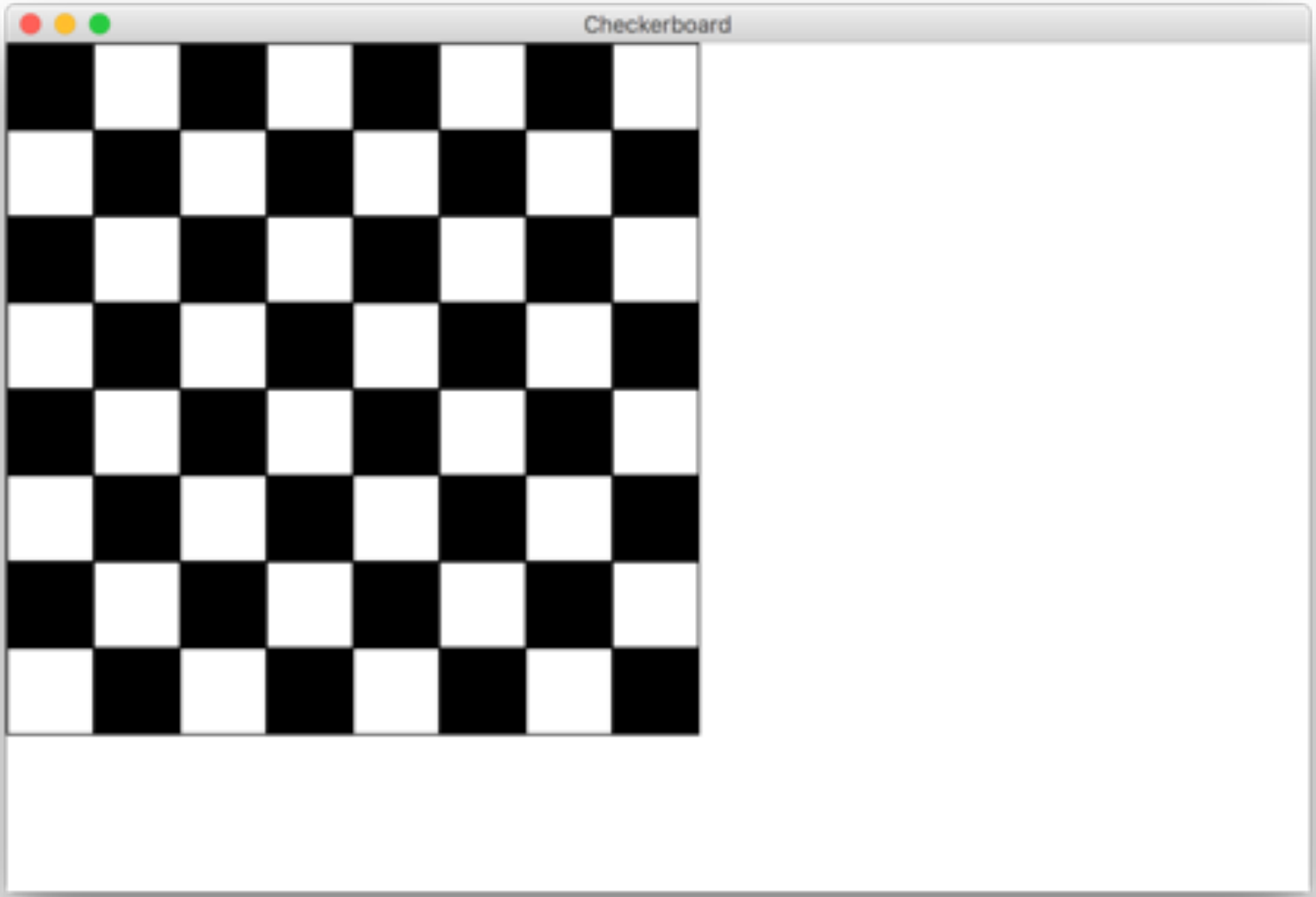


Pedagogy

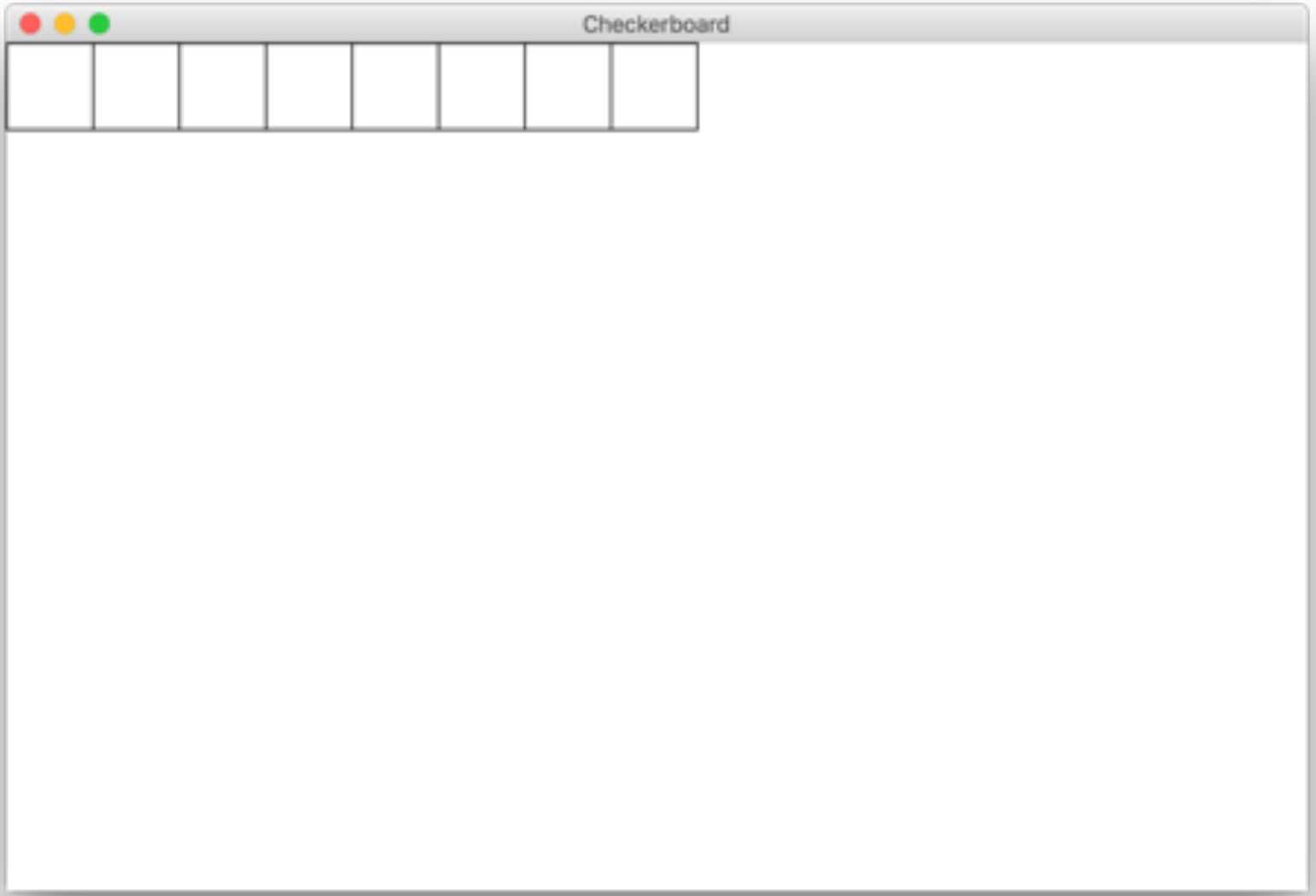

CS106A
localhost:8000/examples/awesome/
CS106A Lectures Assignments Section Handouts Examples Schedule
Programming is Awesome
Programming is Awesome
BY CHRIS PIECH
Graphics are really fantastic in python, especially using the TK library (which is the standard for Python). There are a lot of details, and as such a great way to learn is to look at worked examples.
Programming is Awesome!
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")


Handouts
Examples
General Information
Course Placement
Honor Code
Installing PyCharm
Using Karel in PyCharm
Submitting Assignments
Diagnostic
Graphics Reference

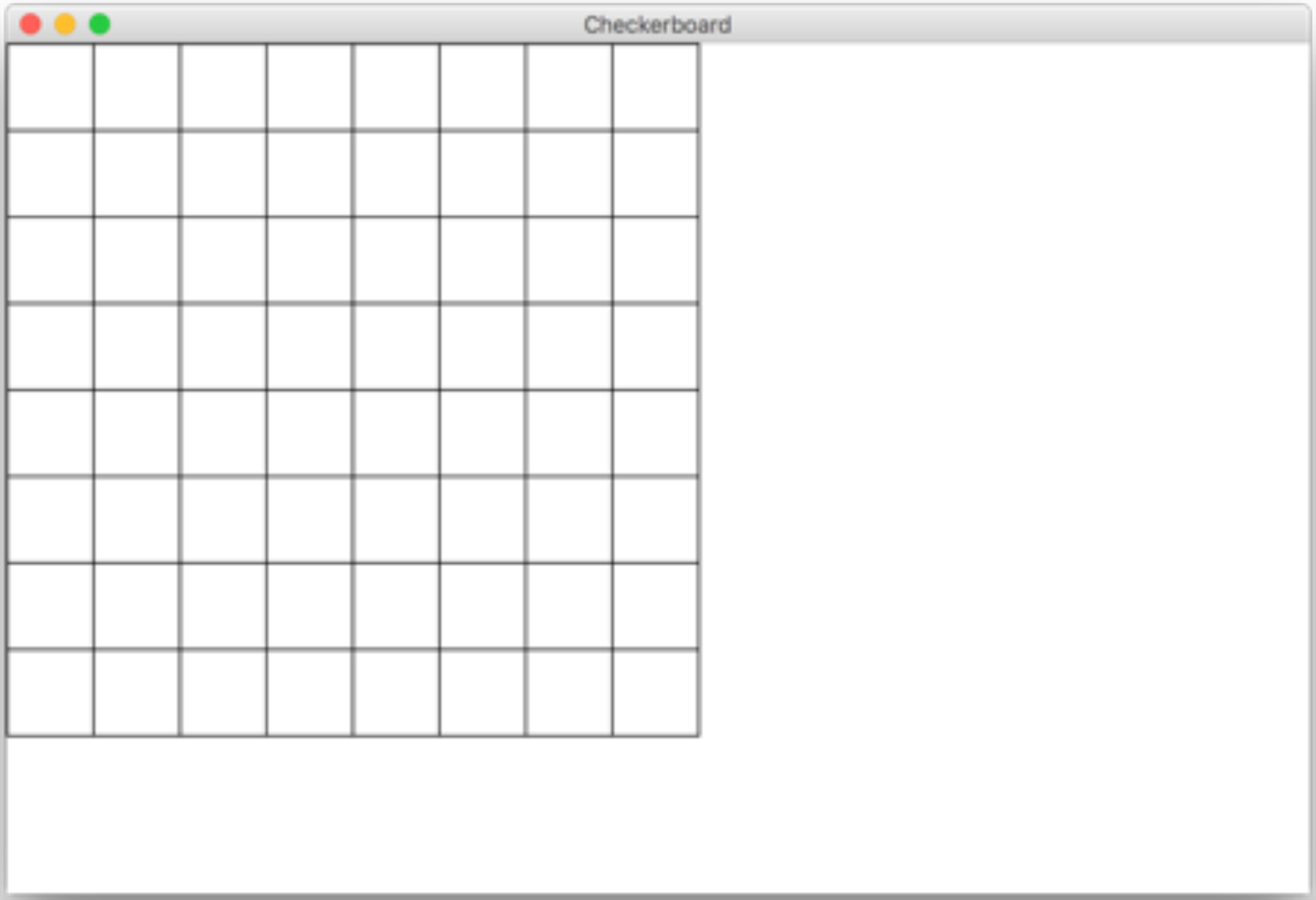
Goal



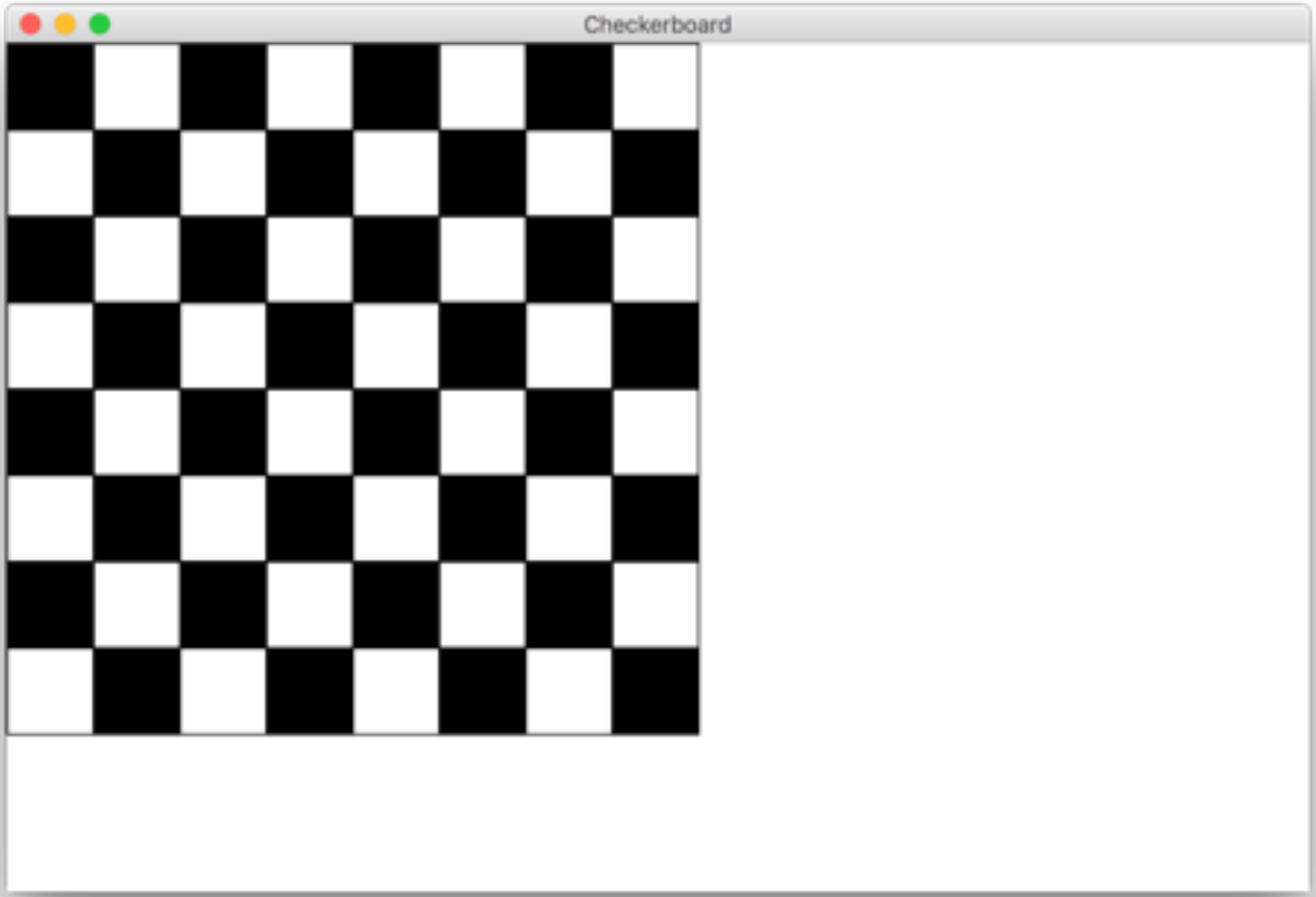
Milestone 1

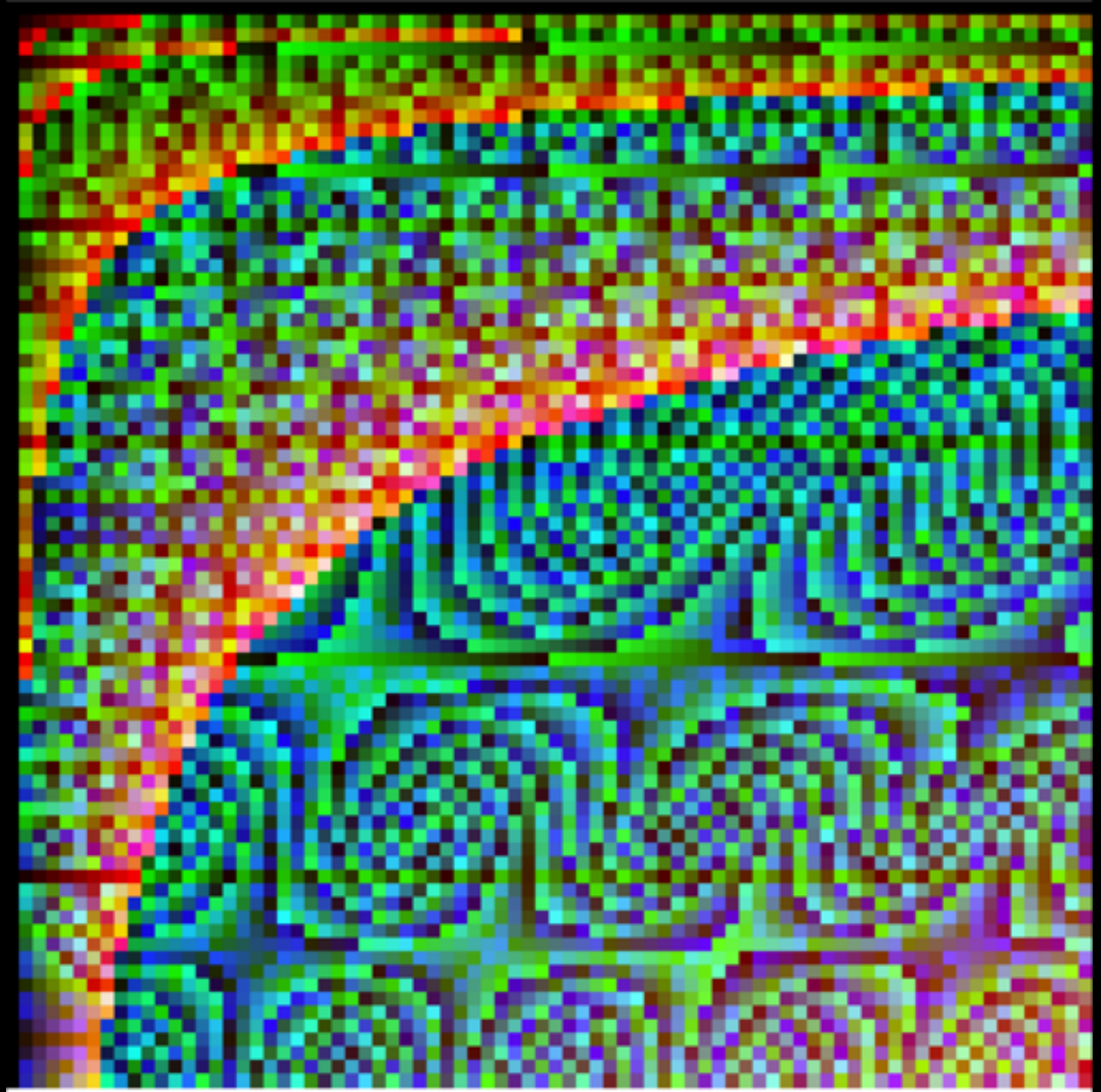


Milestone 2



Milestone 3





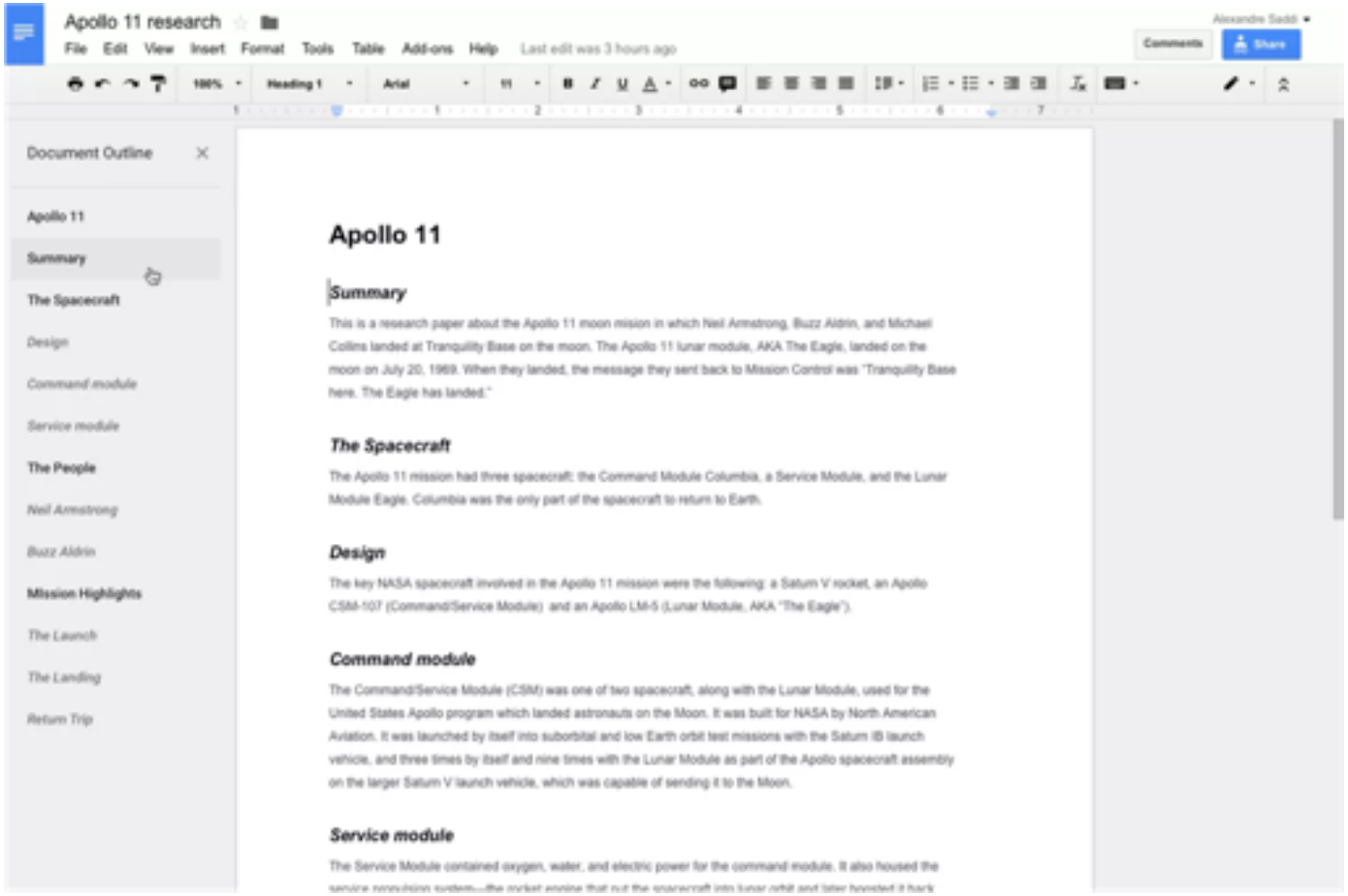
Hold up!

```
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)
```

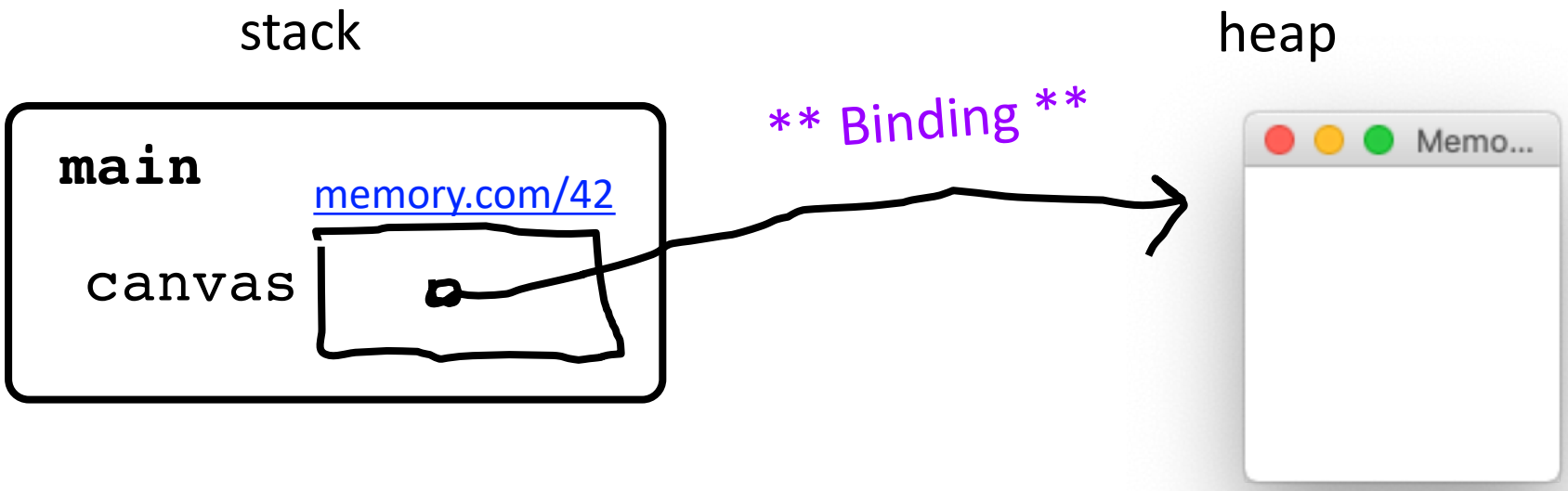
---

stack

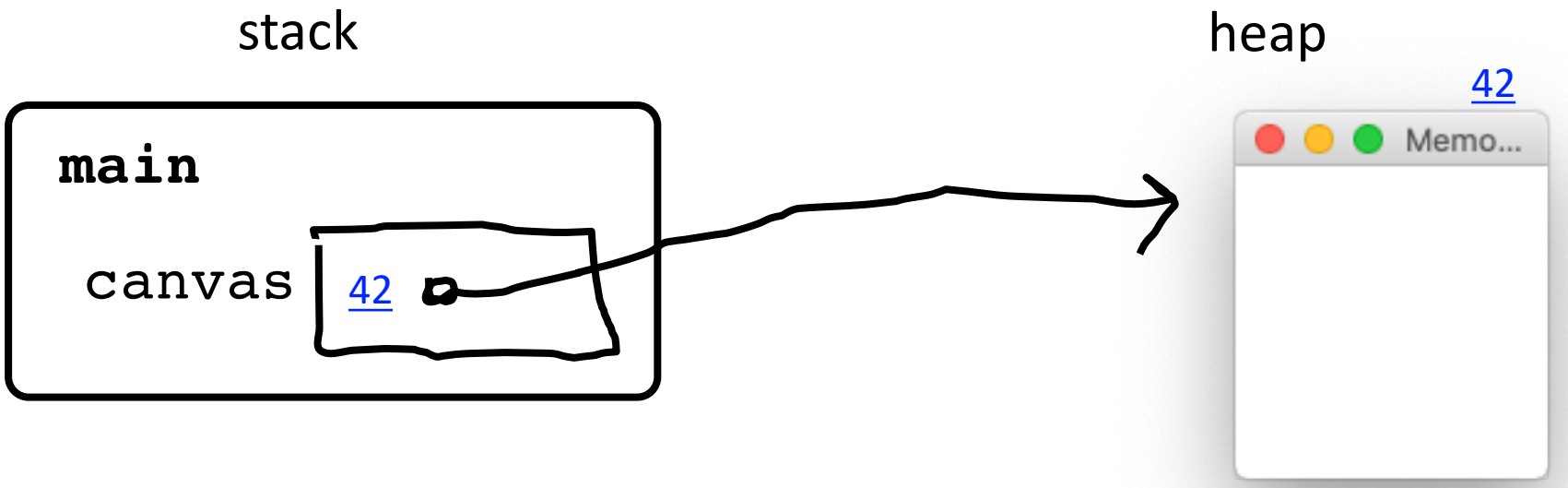
heap

main

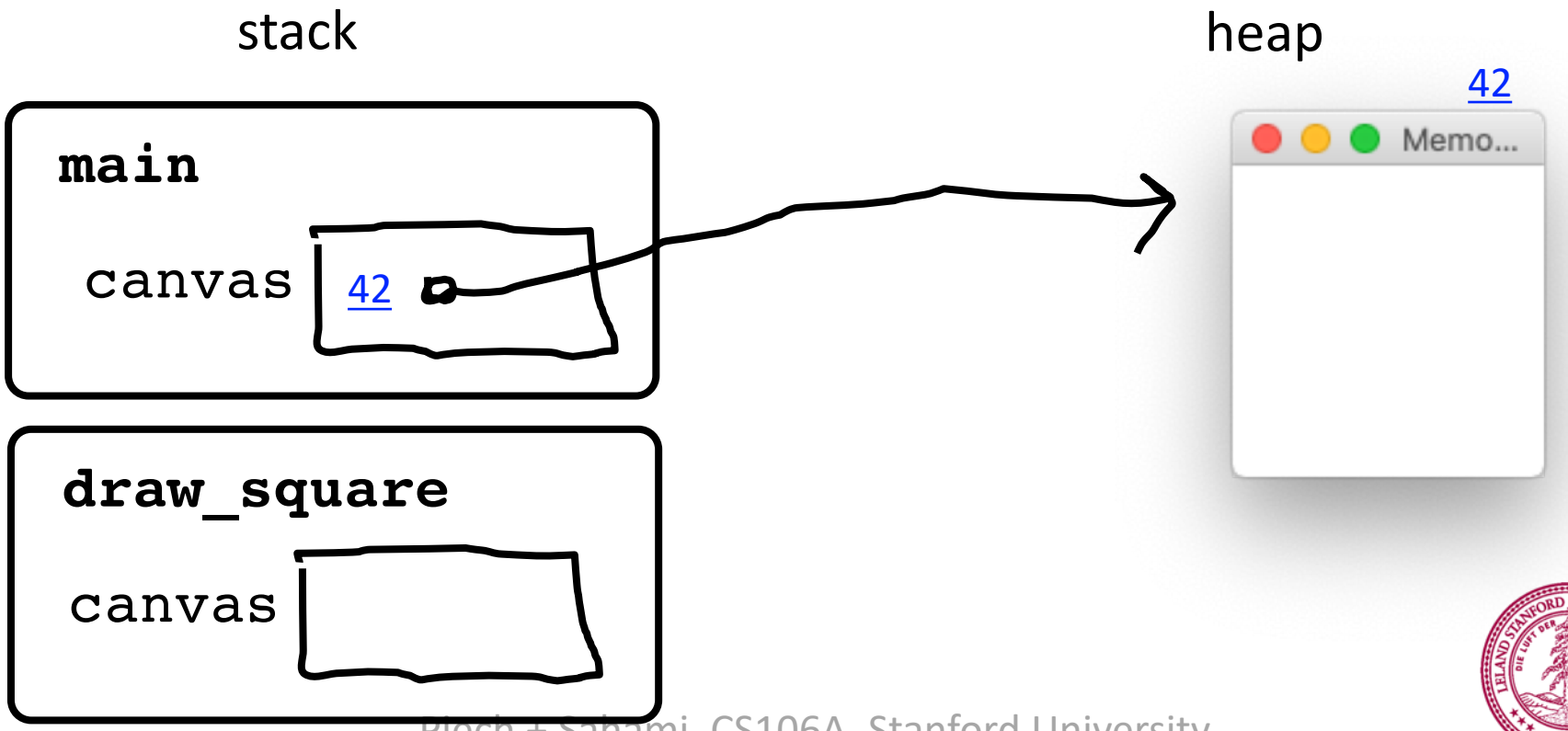
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



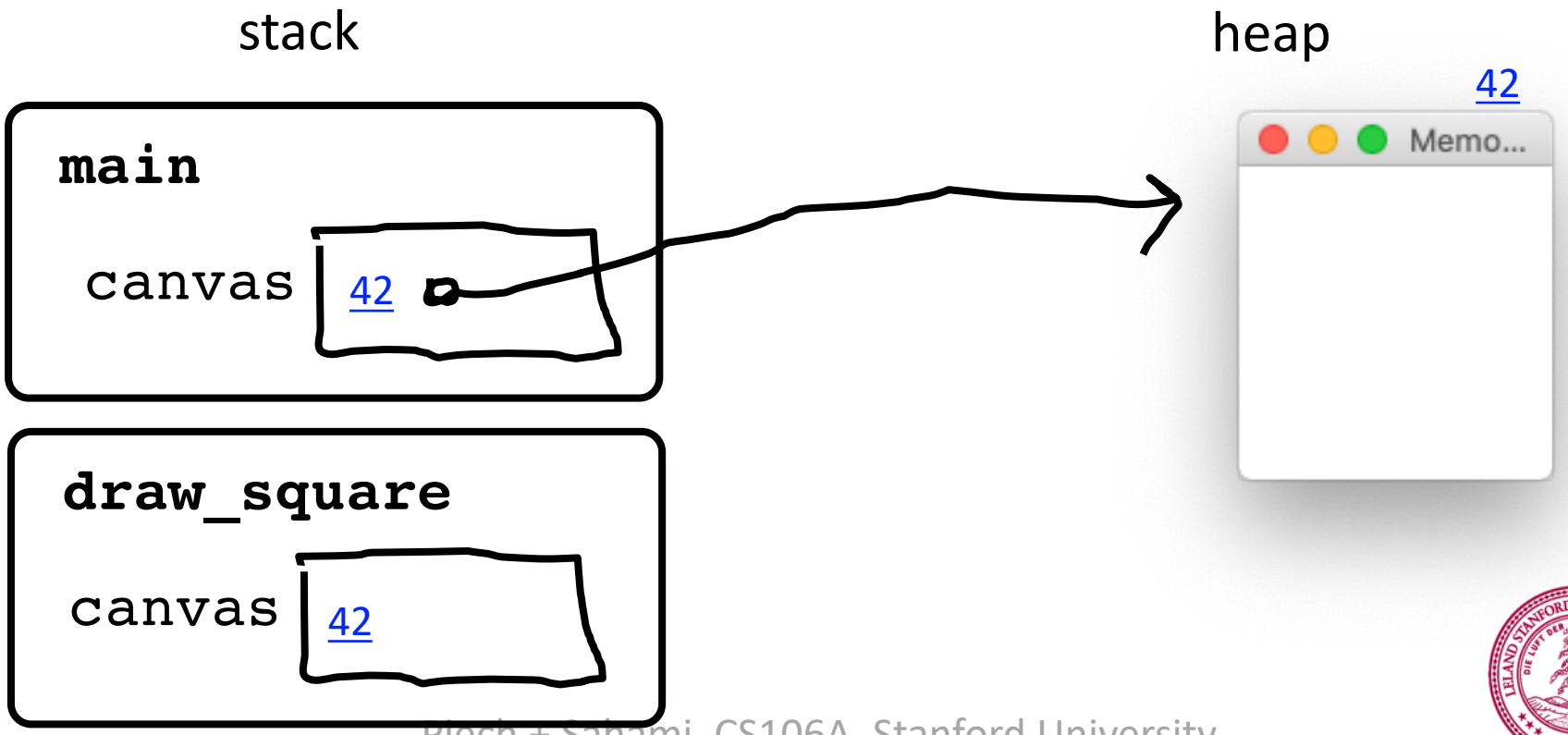
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



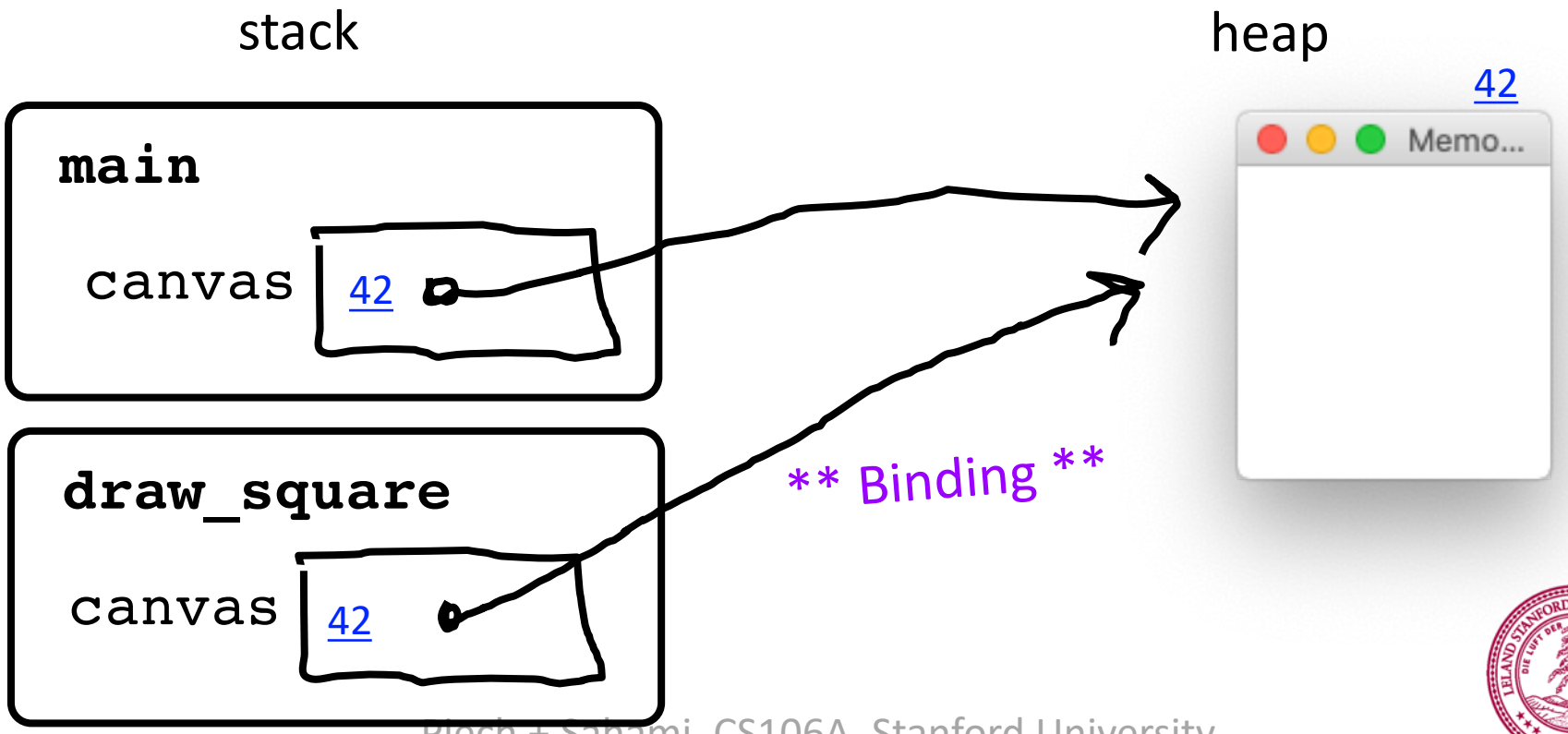
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



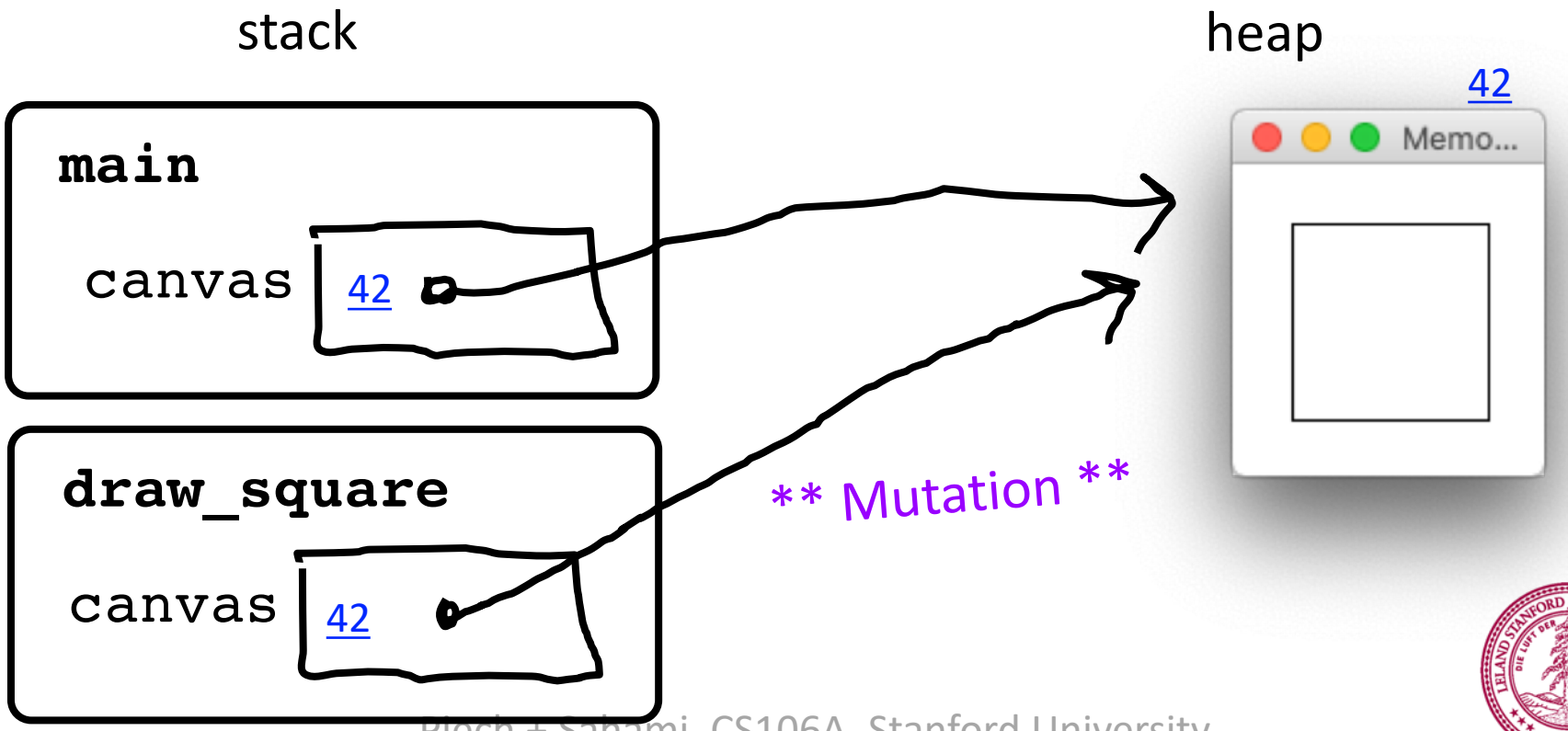
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



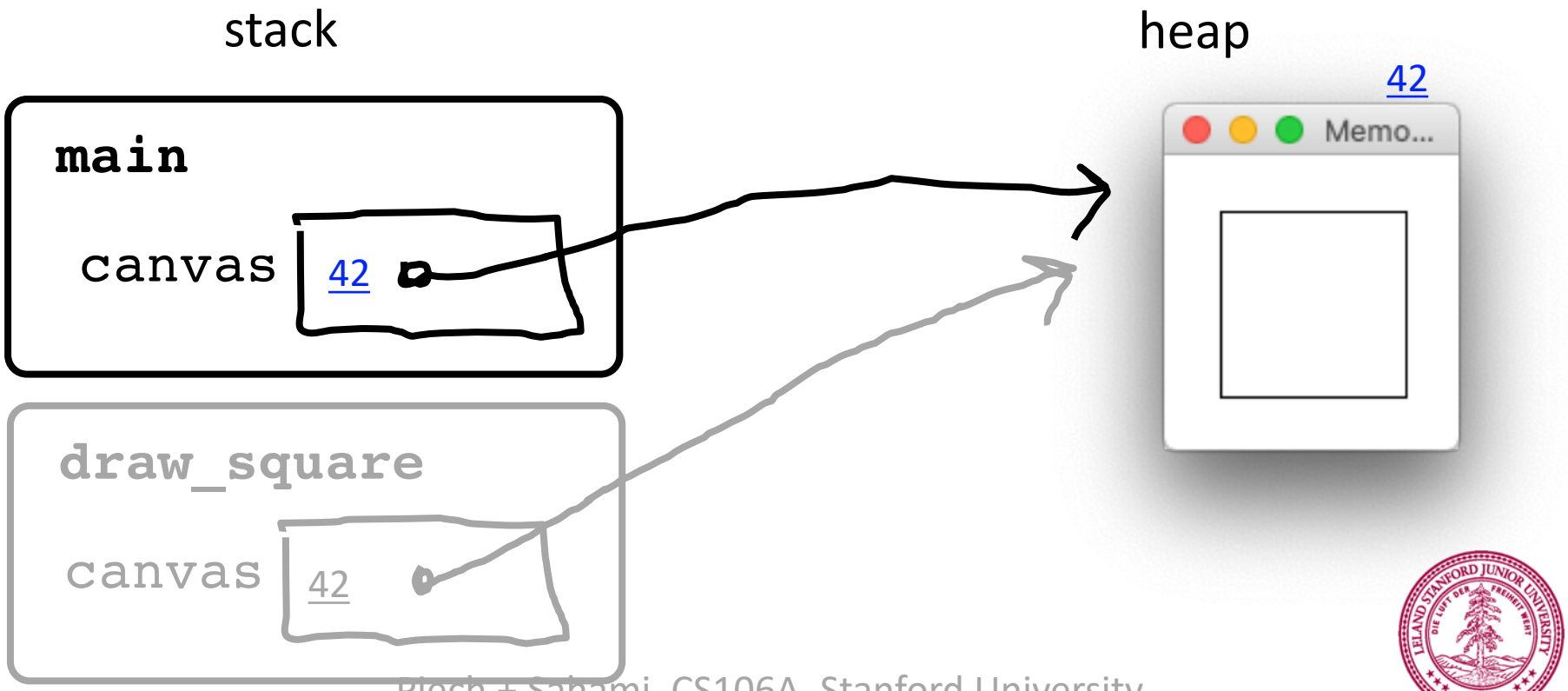
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



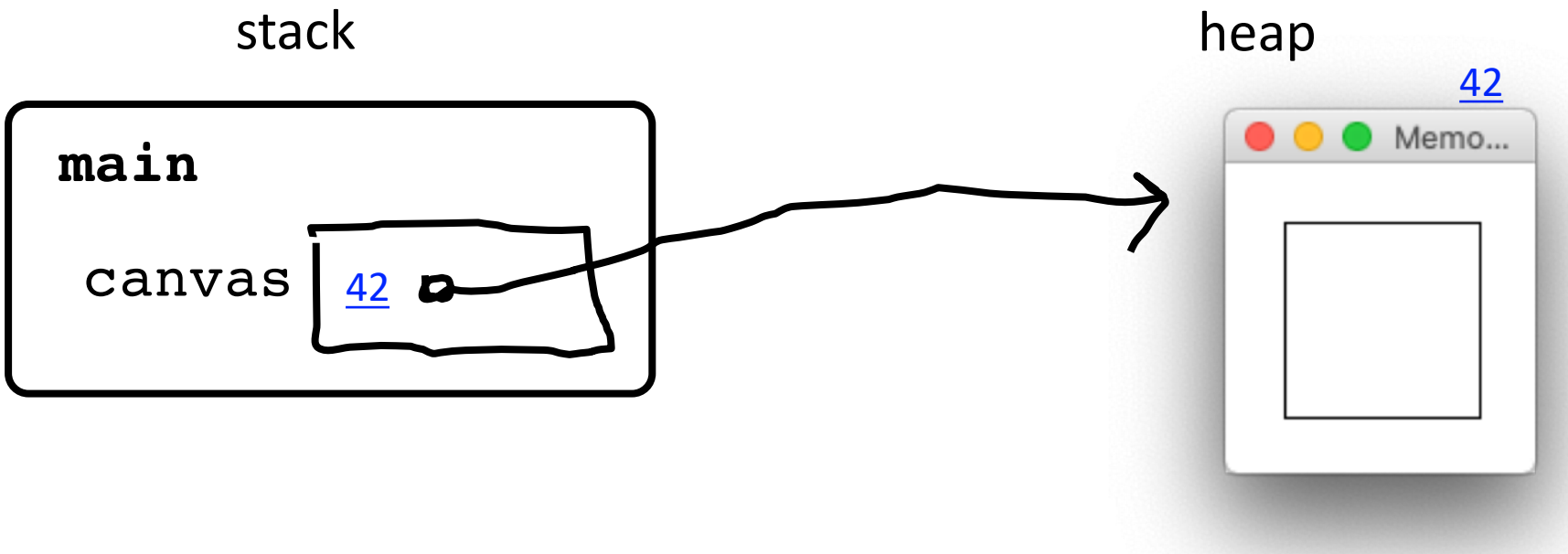
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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)
```



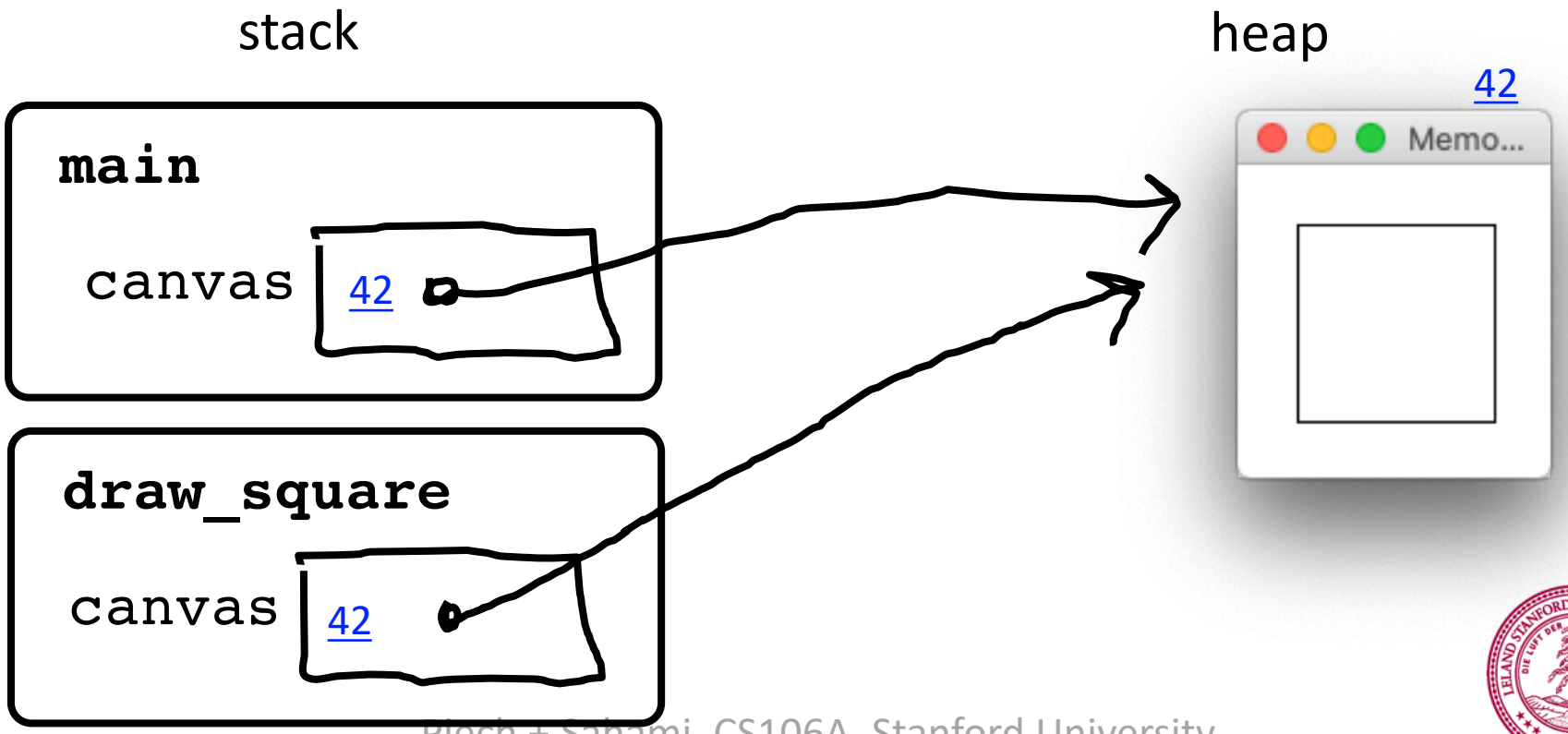
Variables are stored as memory addresses.

Binding is changing memory address.

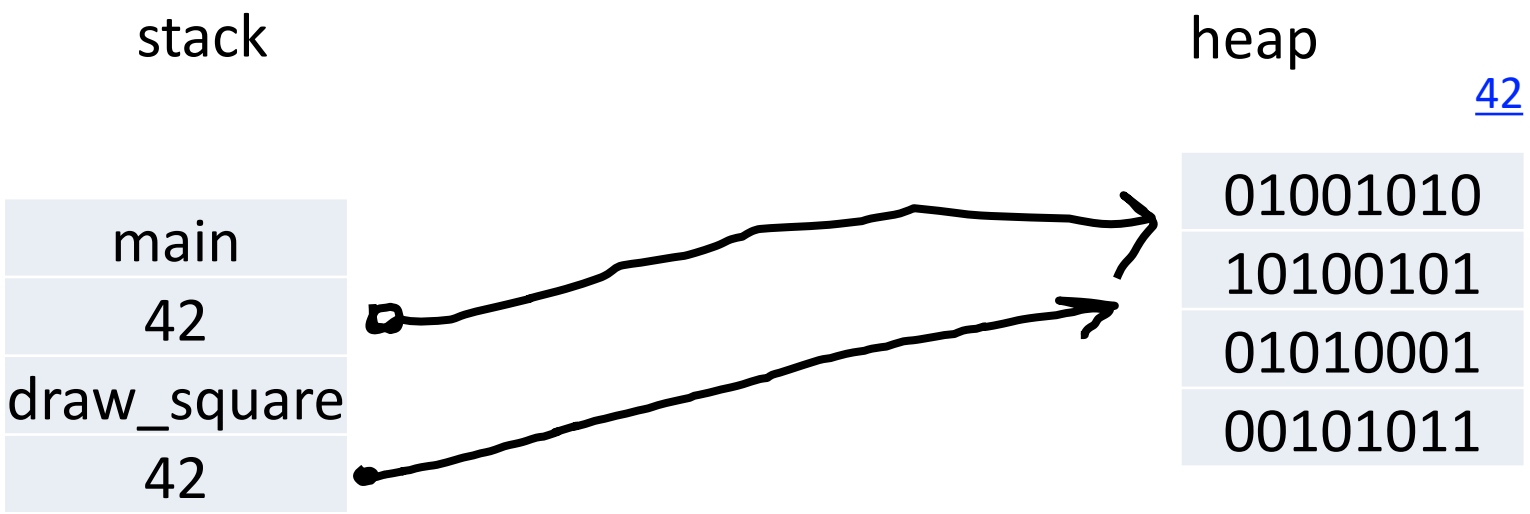
Mutating is editing whats on the other end.

(memory addresses are like memory URLs)

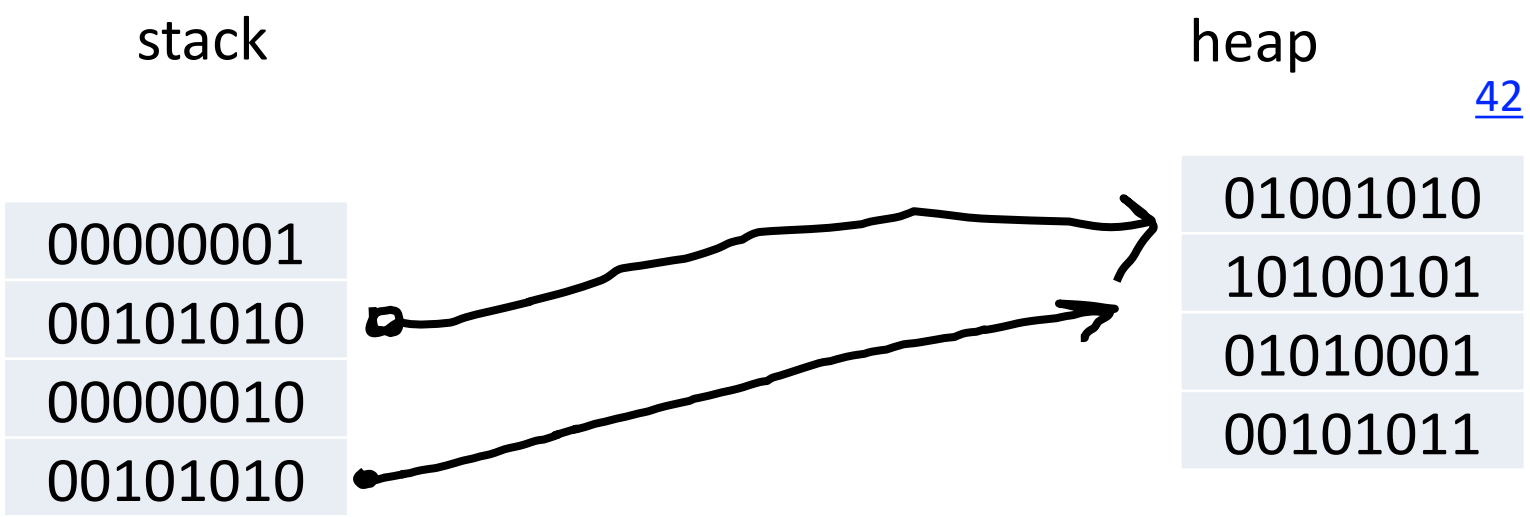
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



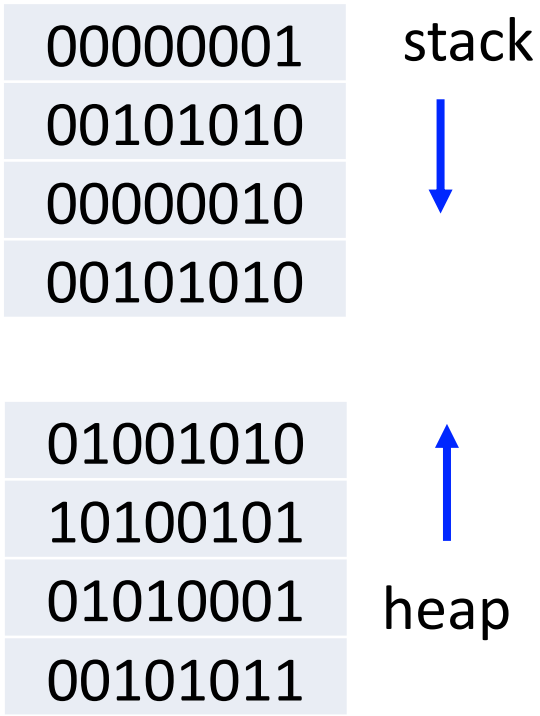
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



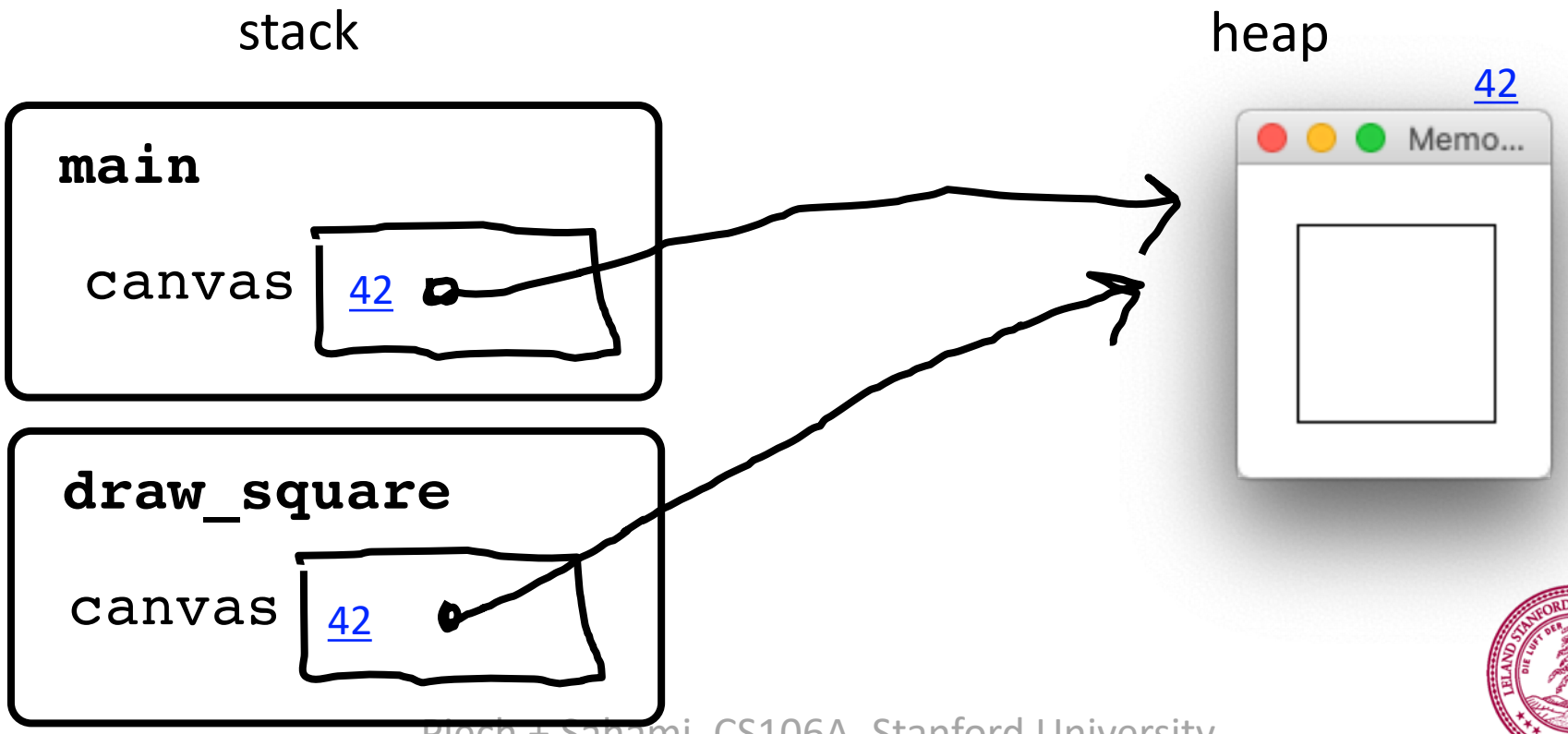
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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



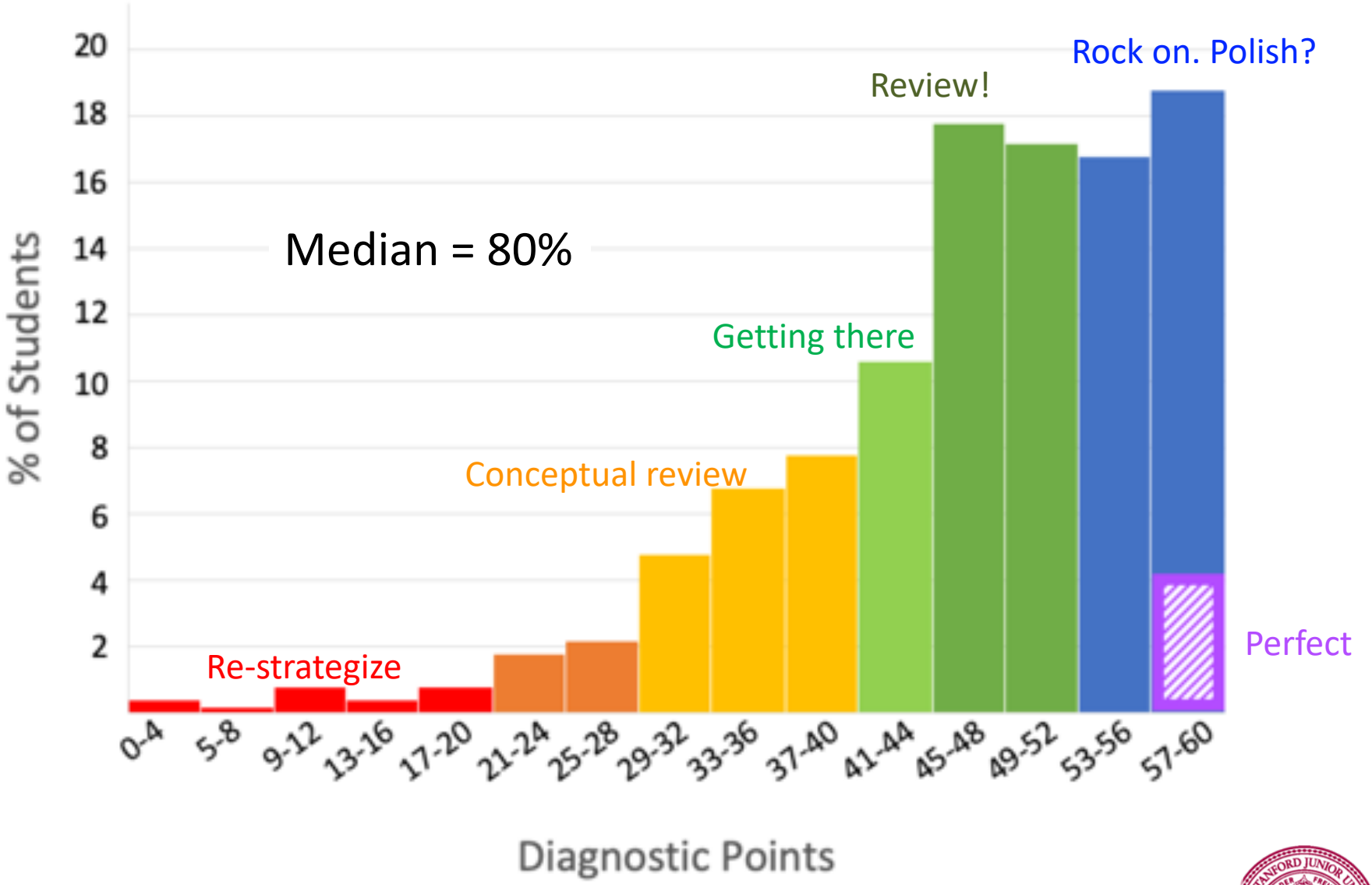
```
def main():
    canvas = make_canvas(…)
    draw_square(canvas)

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)
```



Diagnostic 1



Diagnostic 1

- Mastery is still under your control
  - Feedback is how one gets better
  - Many people have trouble at this point, but hit their learning goals by the end of the quarter
- Grade is still under your control
  - We look at the delta between diagnostics
  - 10% was intentional
- First CS test!