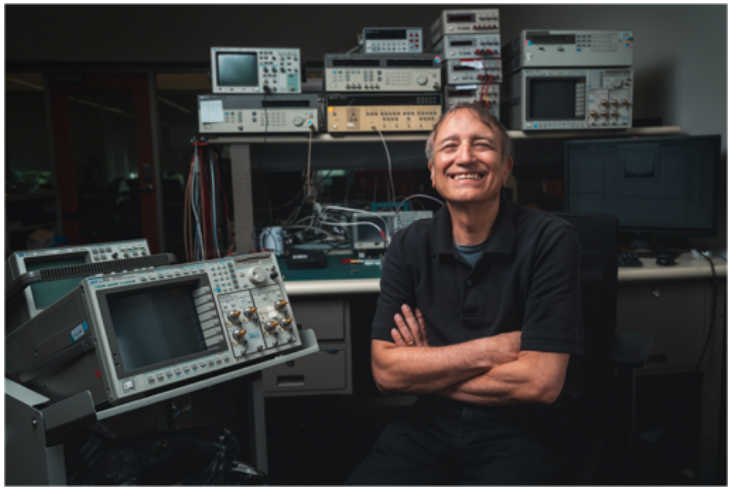
Animation

Chris Piech + Mehran Sahami
CS106A, Stanford University

Turing Award Winner Talks to CS106A



Pixar Night

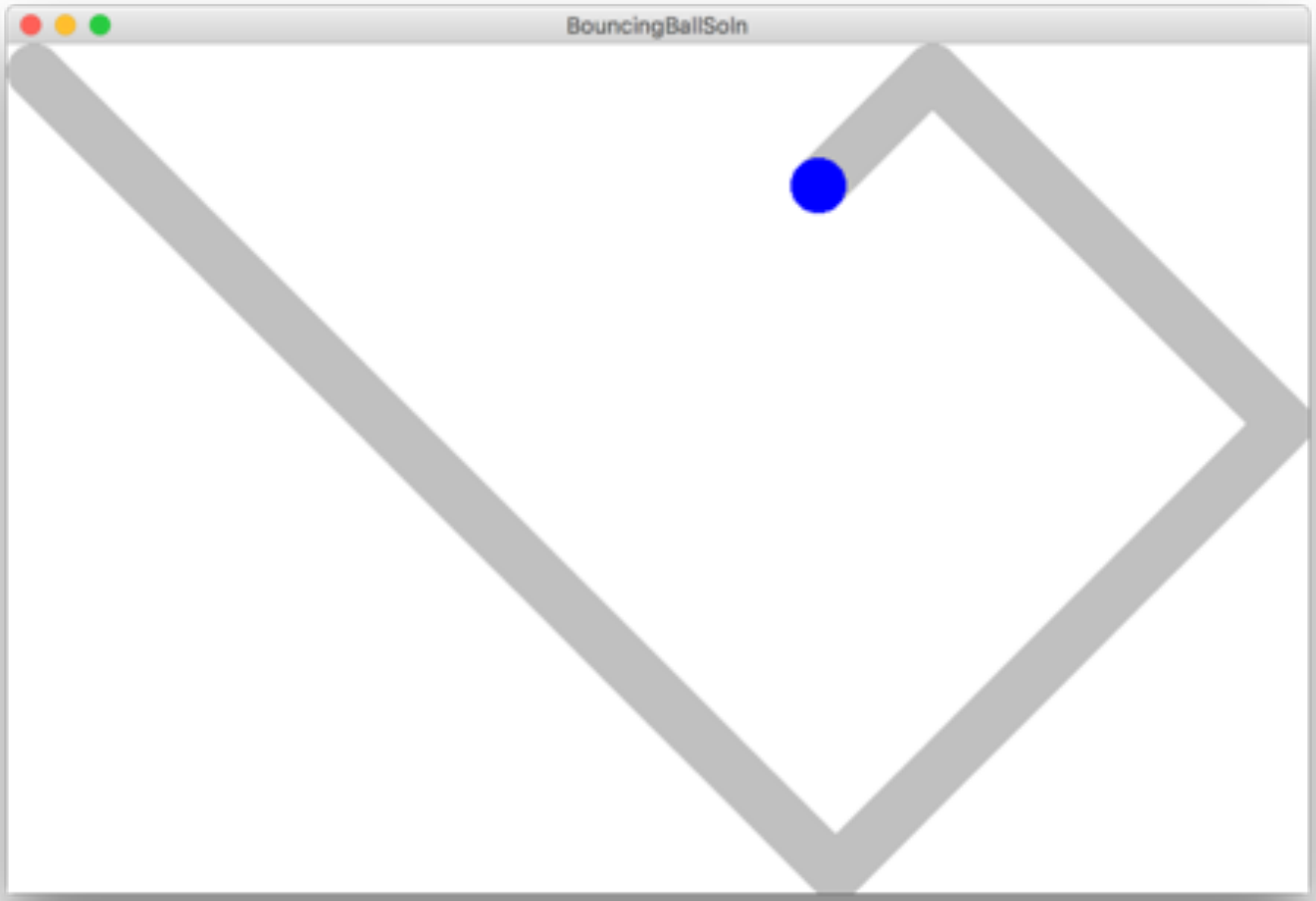
- **Turing Award** is like the nobel prize in CS
- Professor here at Stanford (CS107E, CS348)
- Founding employee at Pixar
- Wrote RenderMan, won 3 Academy Awards
- And just a really wonderful human.
- Thursday **May 7th, 4:30-6pm PDT** over Zoom, and attendance is restricted to students in CS 106A and CS 106B

https://web.stanford.edu/class/cs106a/restricted/pixarnight

Learning Goals

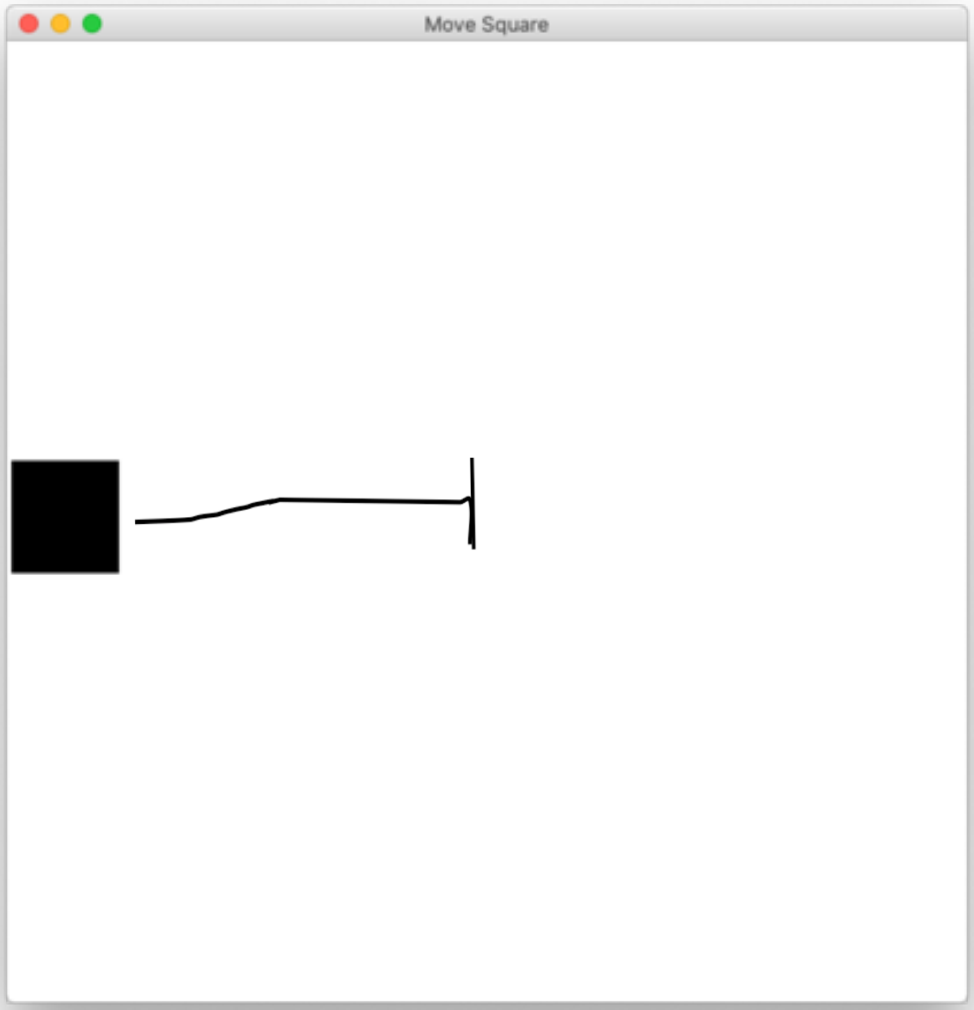
1. Feel more confident debugging
2. Write animated programs

You will be able to write Bouncing Ball



Great foundation

Move to Center



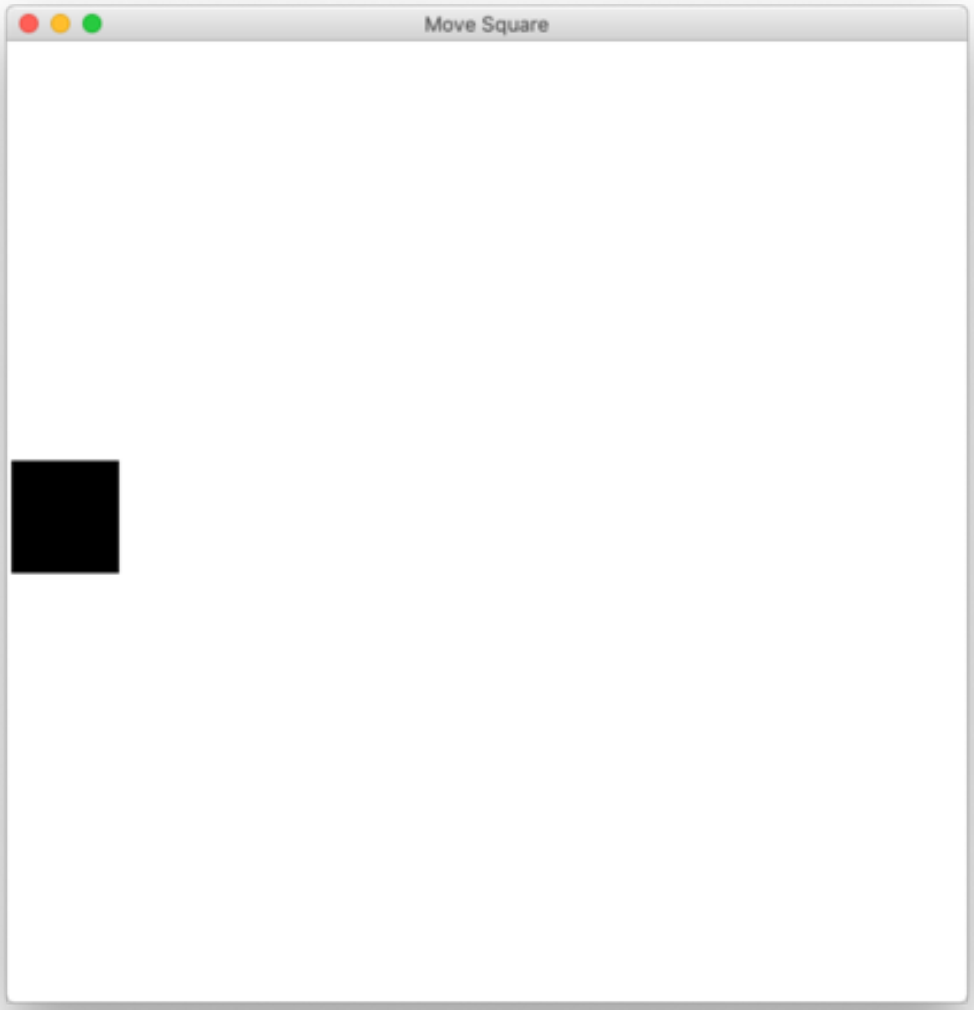
In our last episode...

Graphics from tkinter (aka tk)

```
import tkinter
```

```
# we write this for you, and include it
# in all of your projects!
def make_canvas(width, height, title):
```


Add square



Graphics from tkinter (aka tk)

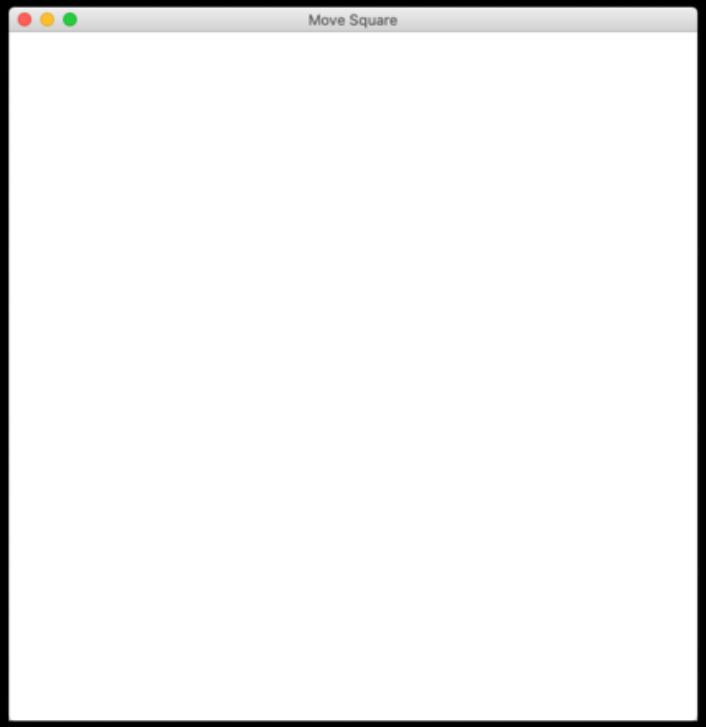
```
import tkinter
```

```
# we write this for you, and include it
# in all of your projects!
def make_canvas(width, height, title):
```

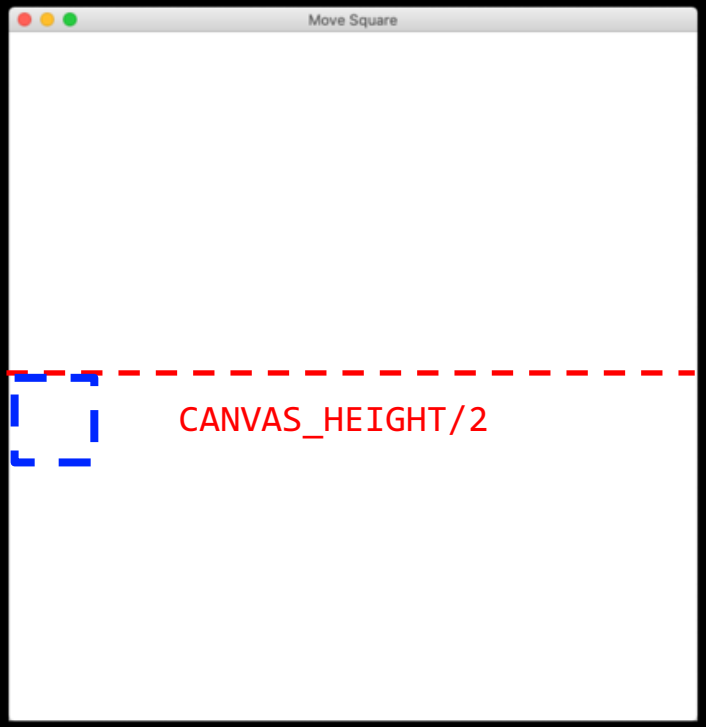

```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```


```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```

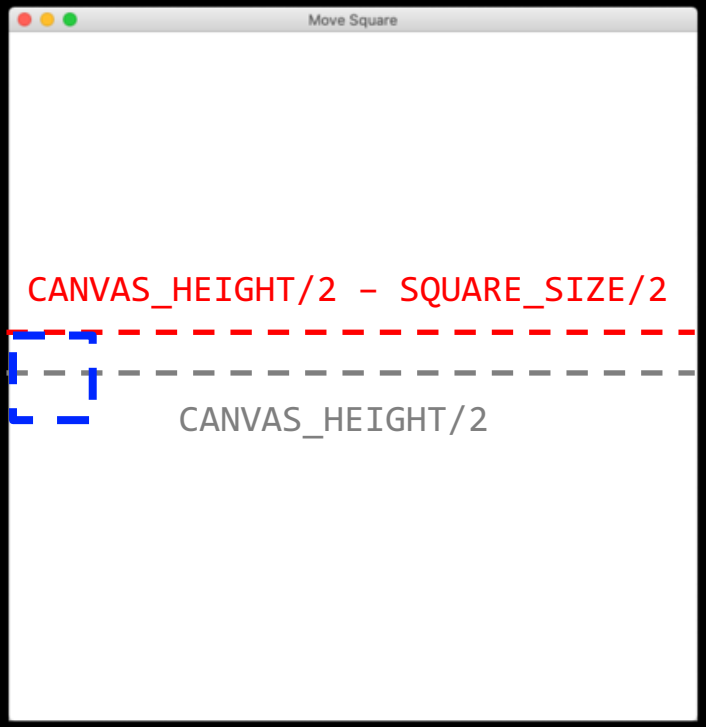

```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



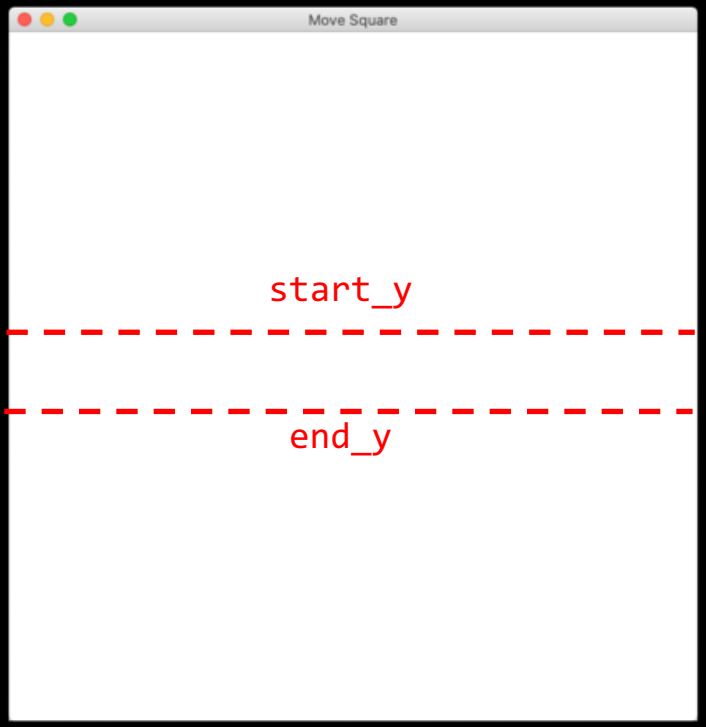
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



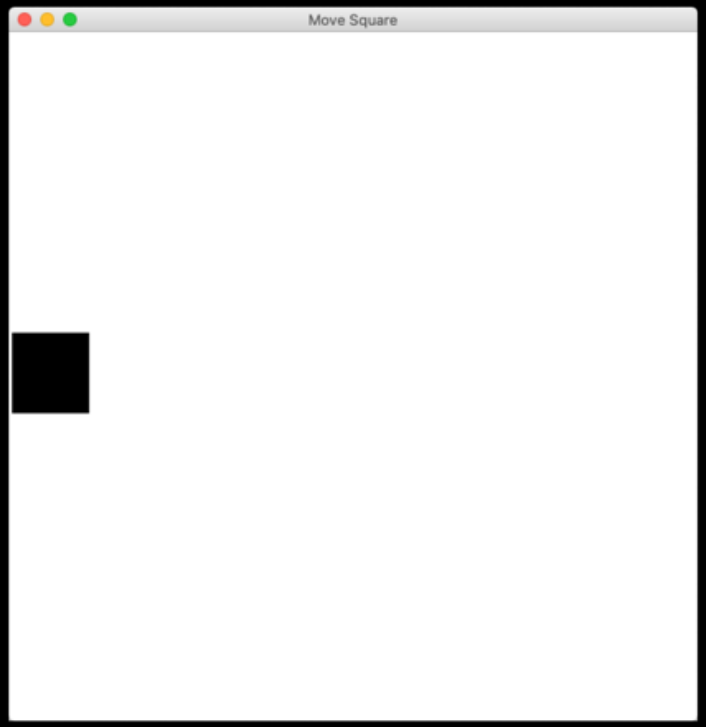
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```

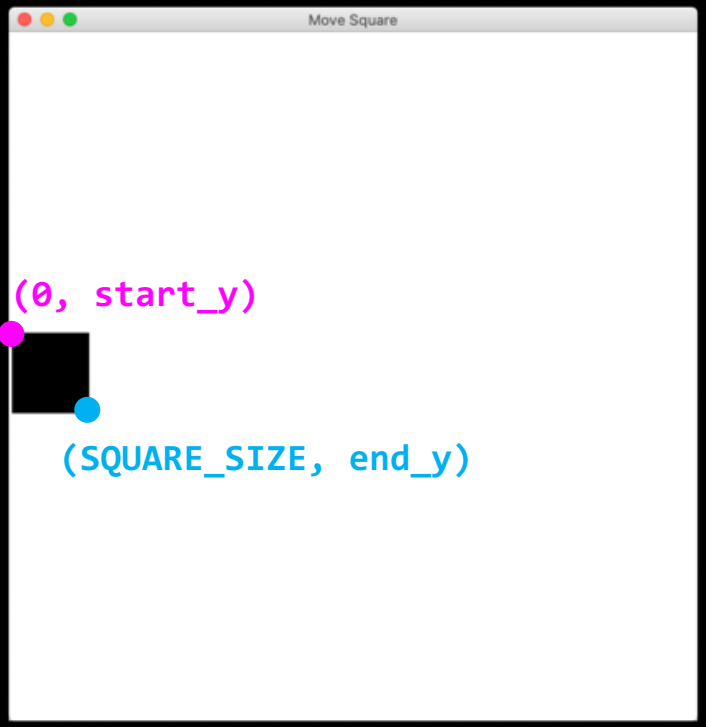


```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



Some “heavy duty” variables allow you to call functions on them

```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```

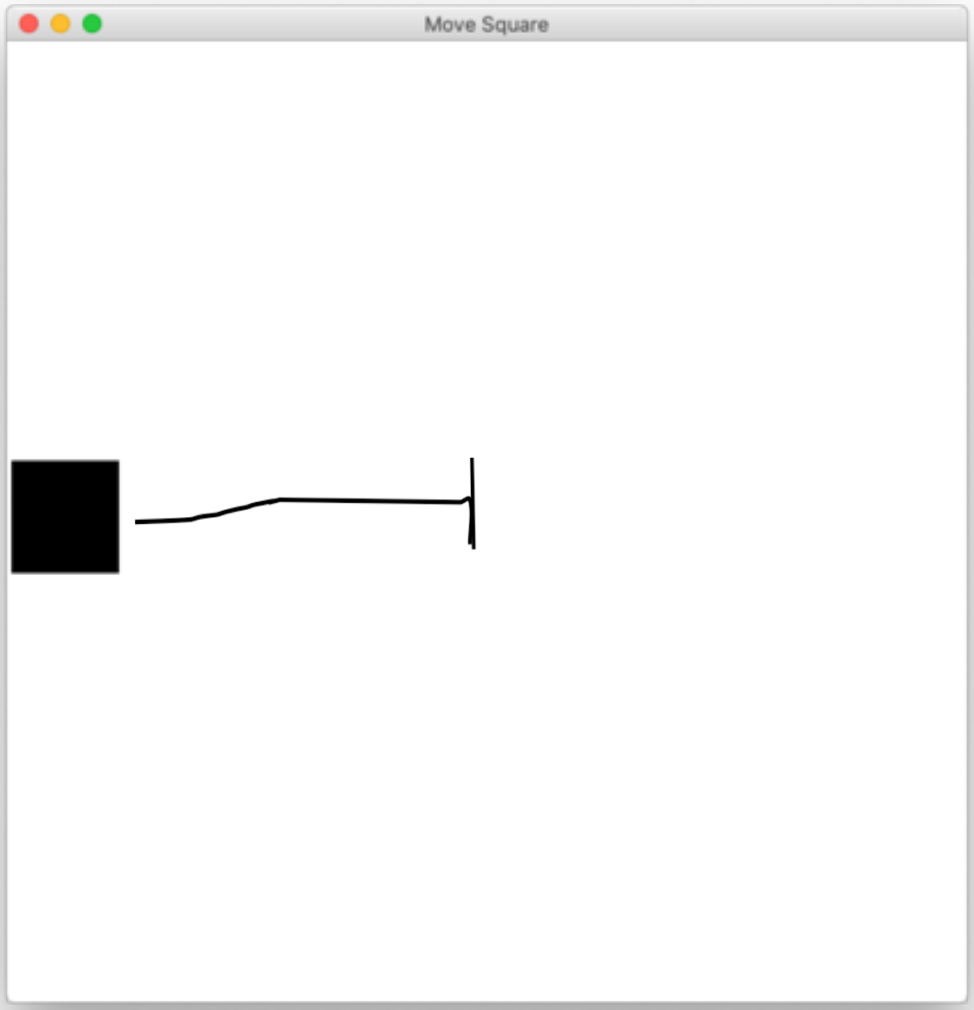


You're now all graphics programmers!

End review...

How do movies or games animate?

Move to Center



* That’s not quite toy story, but it is a start…

Animation Loop

```
def main():
    # setup

    while True:
        # update world

        # pause
        time.sleep(DELAY)
```


Animation Loop

```
def main():
    # setup

    while True:
        # update world

        # pause
        time.sleep(DELAY)
```

Make all the variables you need.

Animation Loop

```
def main():
    # setup

    while True:
        # update world

        # pause
        time.sleep(DELAY)
```

The animation loop is a repetition of heartbeats

Animation Loop

```
def main():
    # setup

    while True:
        # update world

        # pause
        time.sleep(DELAY)
```

Each heart-beat, update the world forward one frame

Animation Loop

```
def main():
    # setup

    while True:
        # update world

        # pause
        time.sleep(DELAY)
```

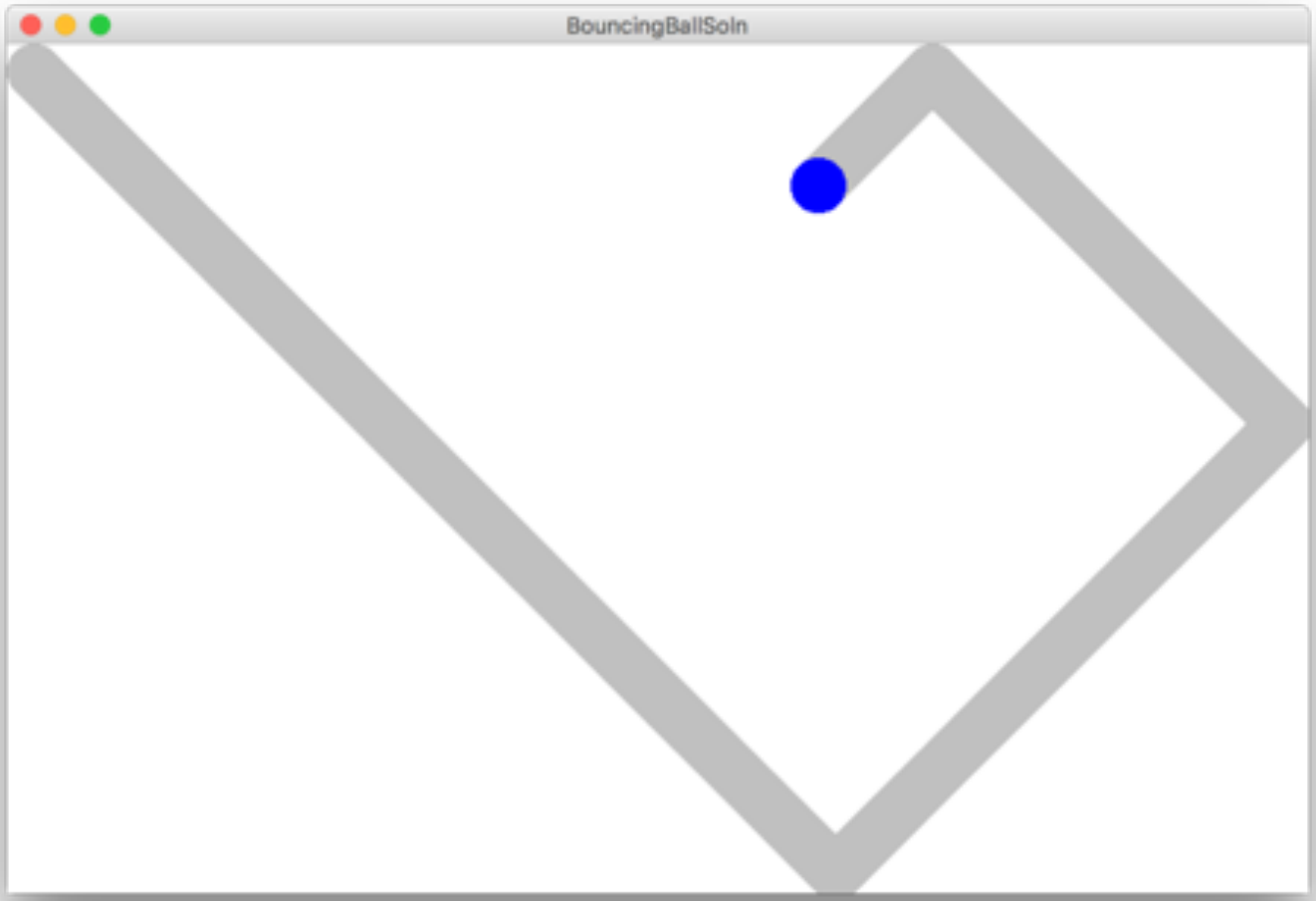
If you don't pause, humans won't be able to see it

Move To Center

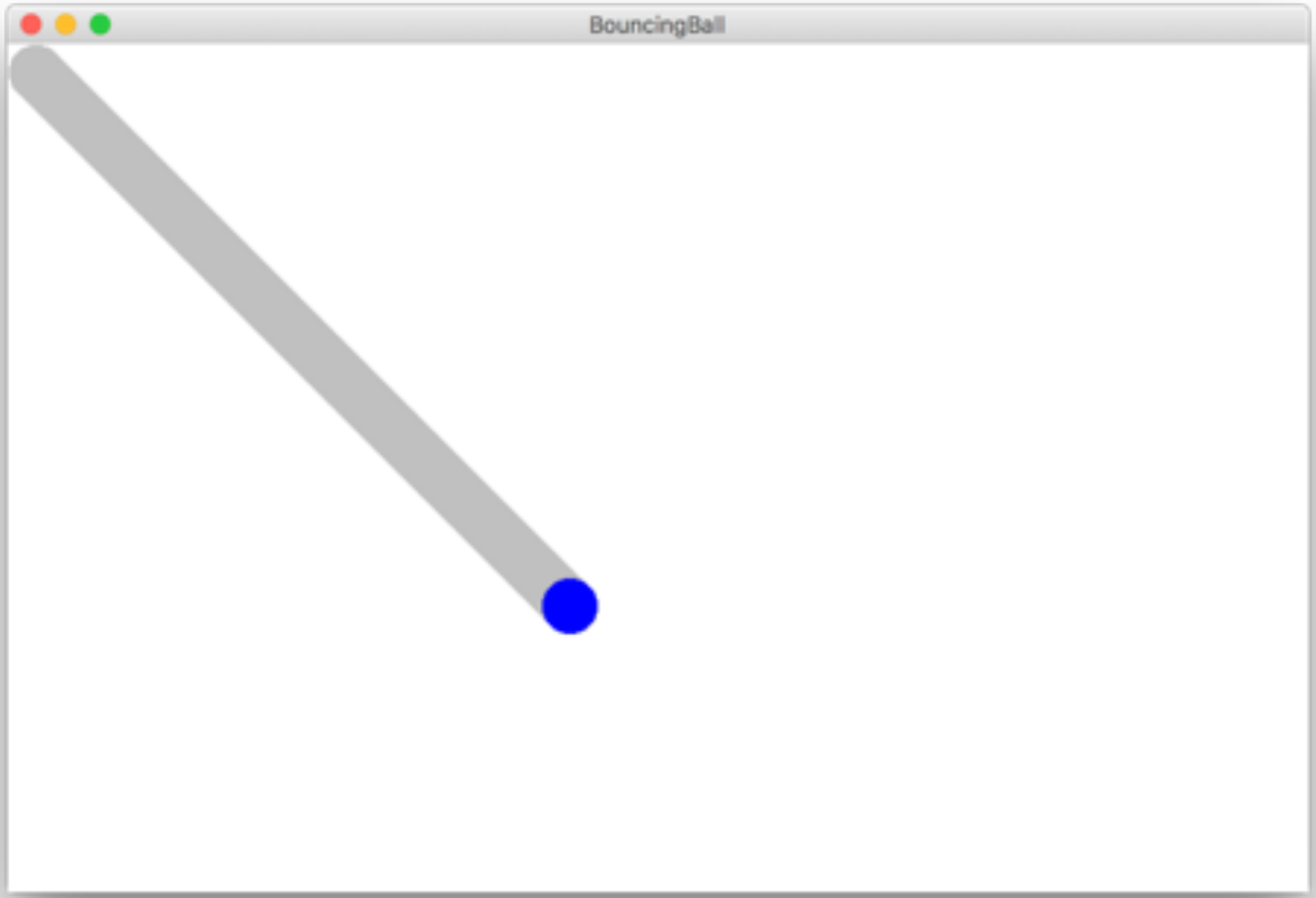
```
def main():
    # setup
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT)
    r = canvas.create_rectangle(0, 0, 100, 100)
    while not is_past_center(canvas, r):
        # update world
        canvas.move(r, 1, 0)
        canvas.update()
        # pause
        time.sleep(DELAY)
```

We are ready…

Bouncing Ball

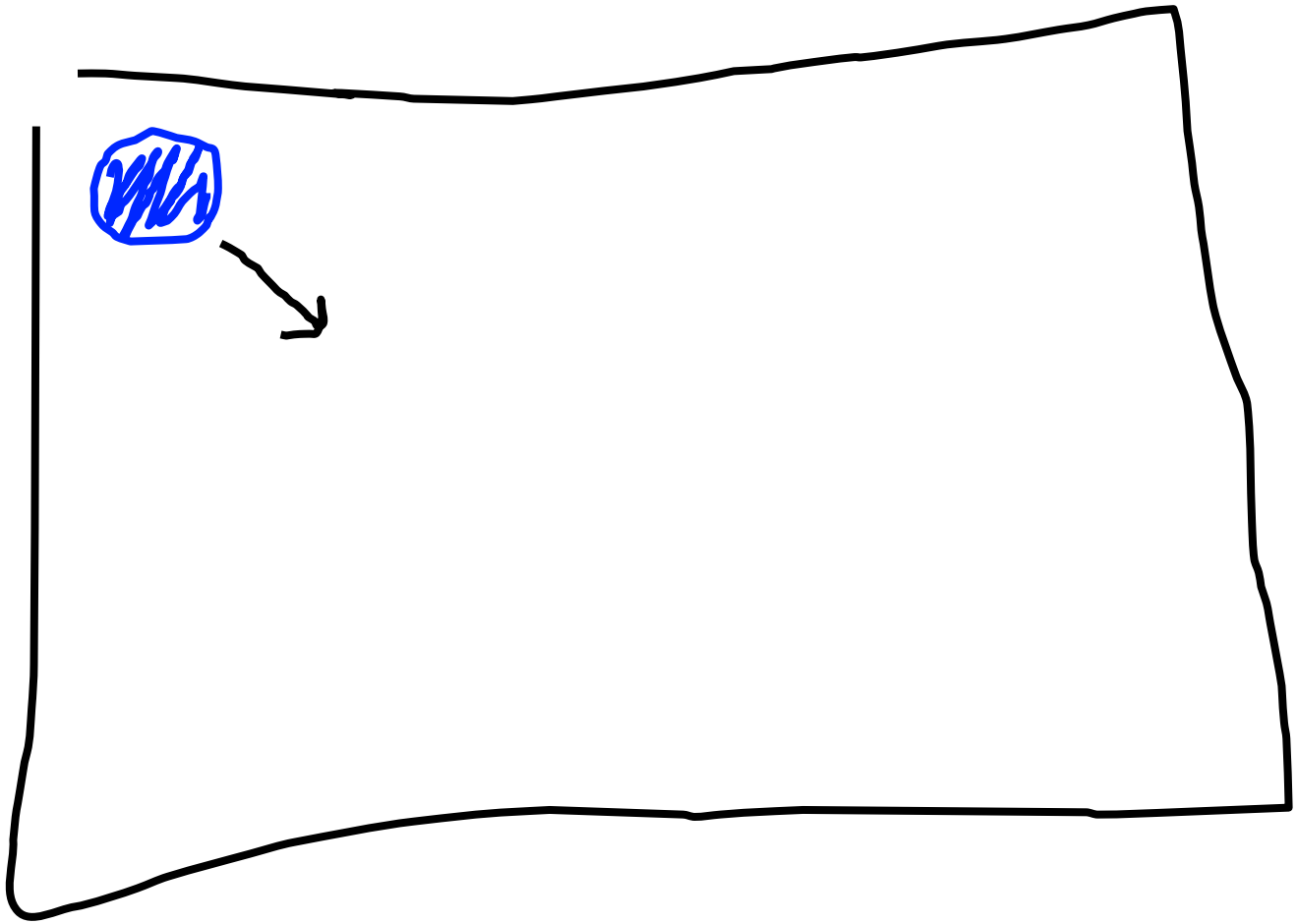


Milestone #1



Bouncing Ball

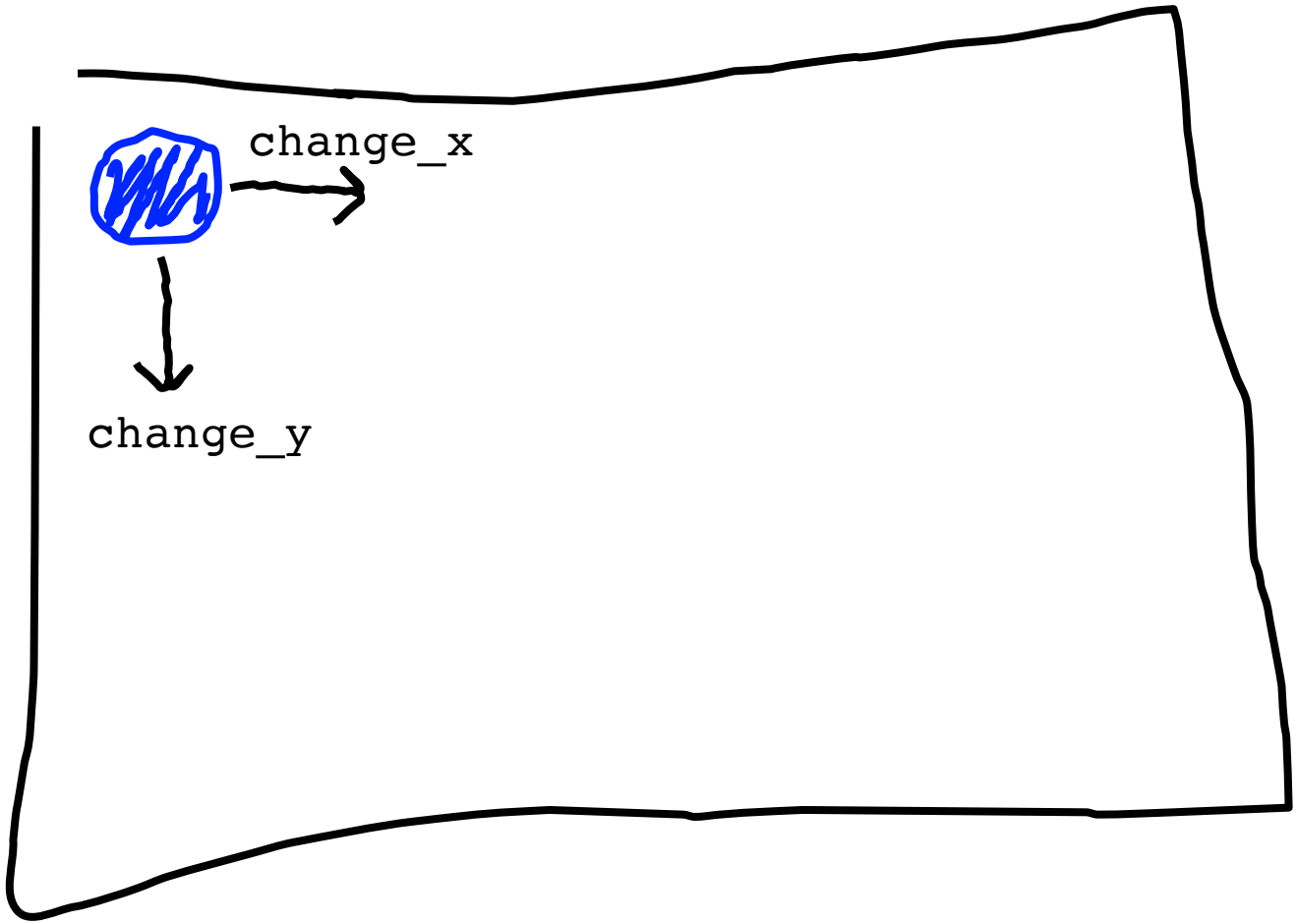
First heartbeat



Key variable: how much the ball position change each heartbeat?

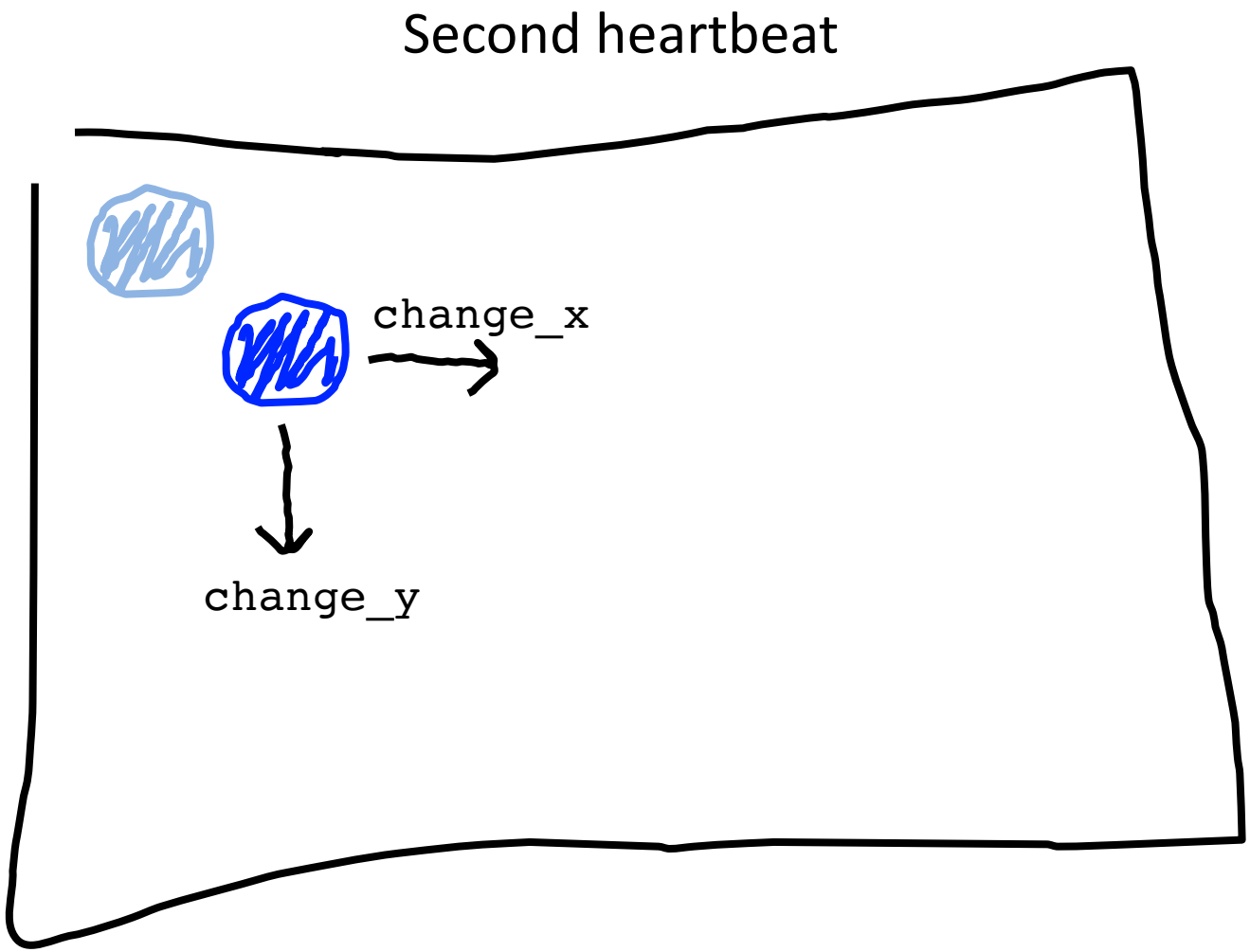
Bouncing Ball

First heartbeat

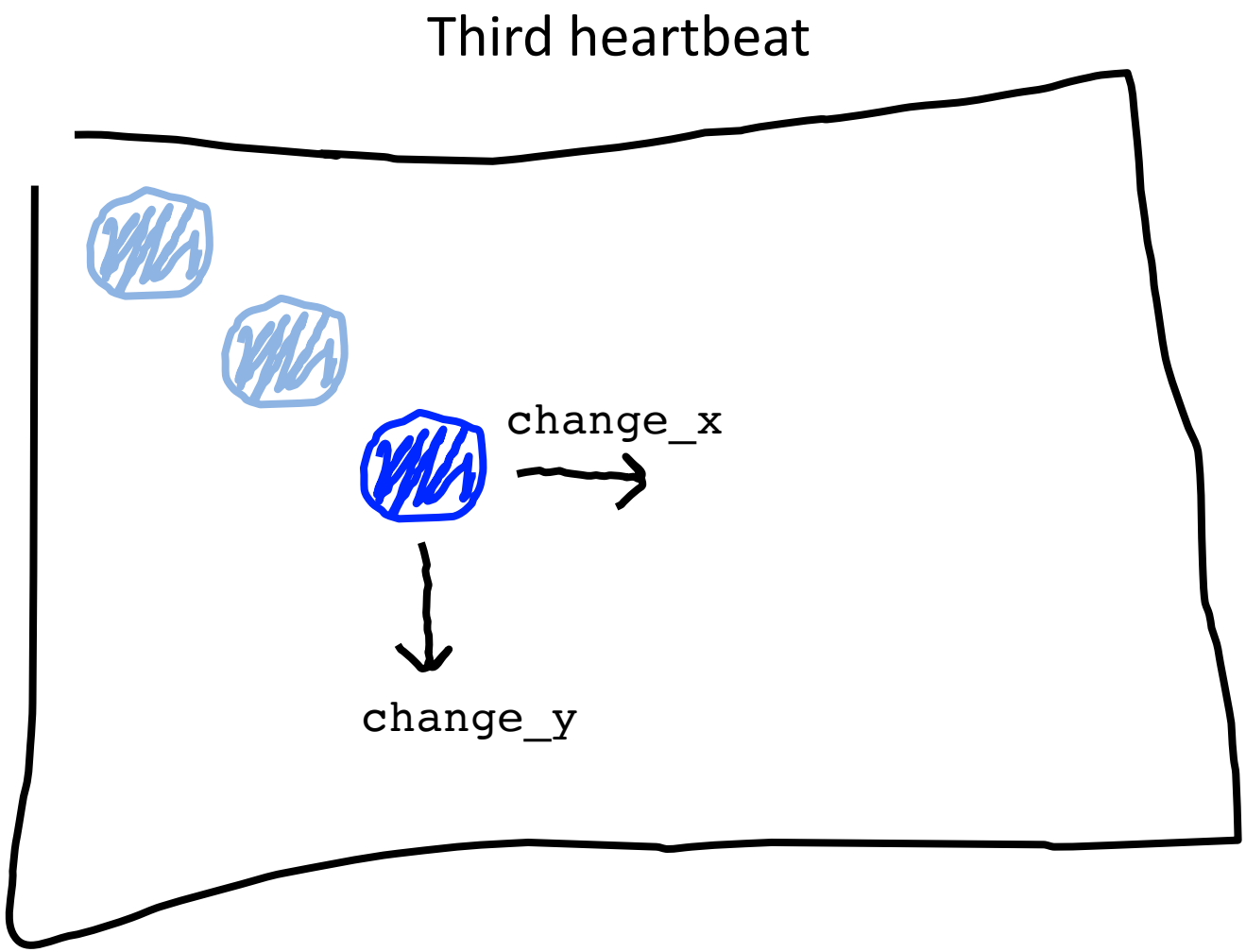


The **`move`** function takes in
a change in x and a change in y

Bouncing Ball

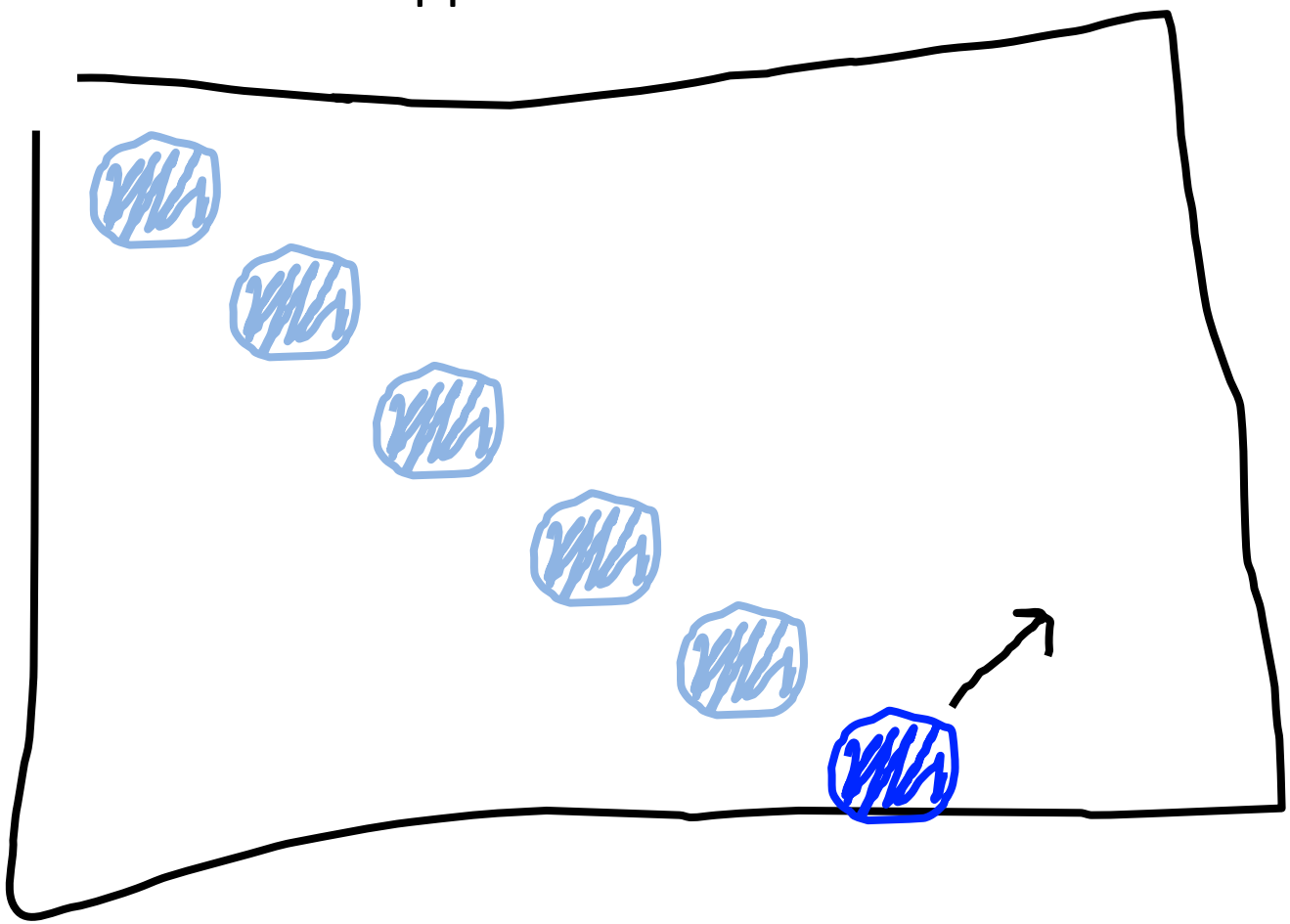


Bouncing Ball



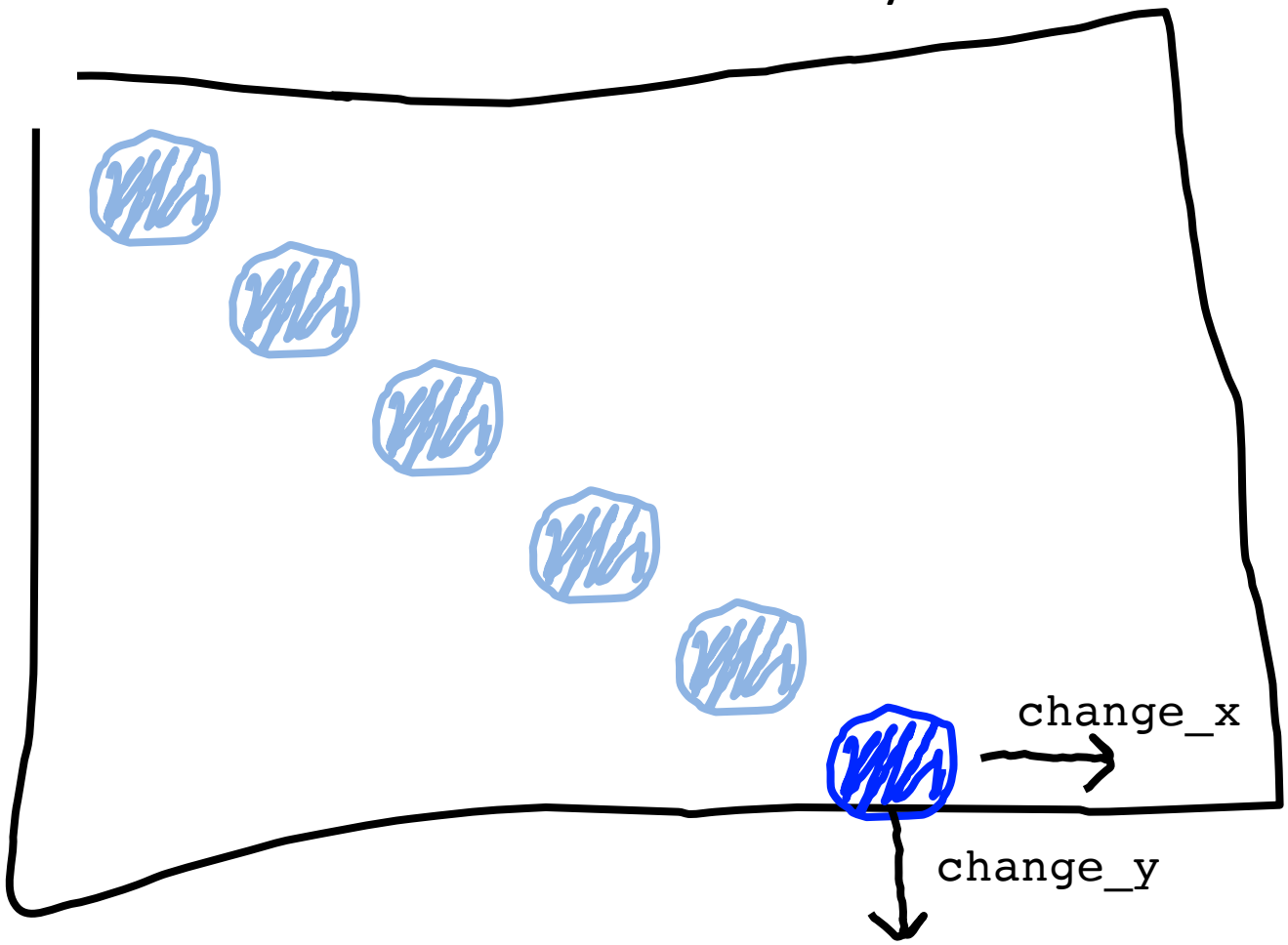
Bouncing Ball

What happens when we hit a wall?



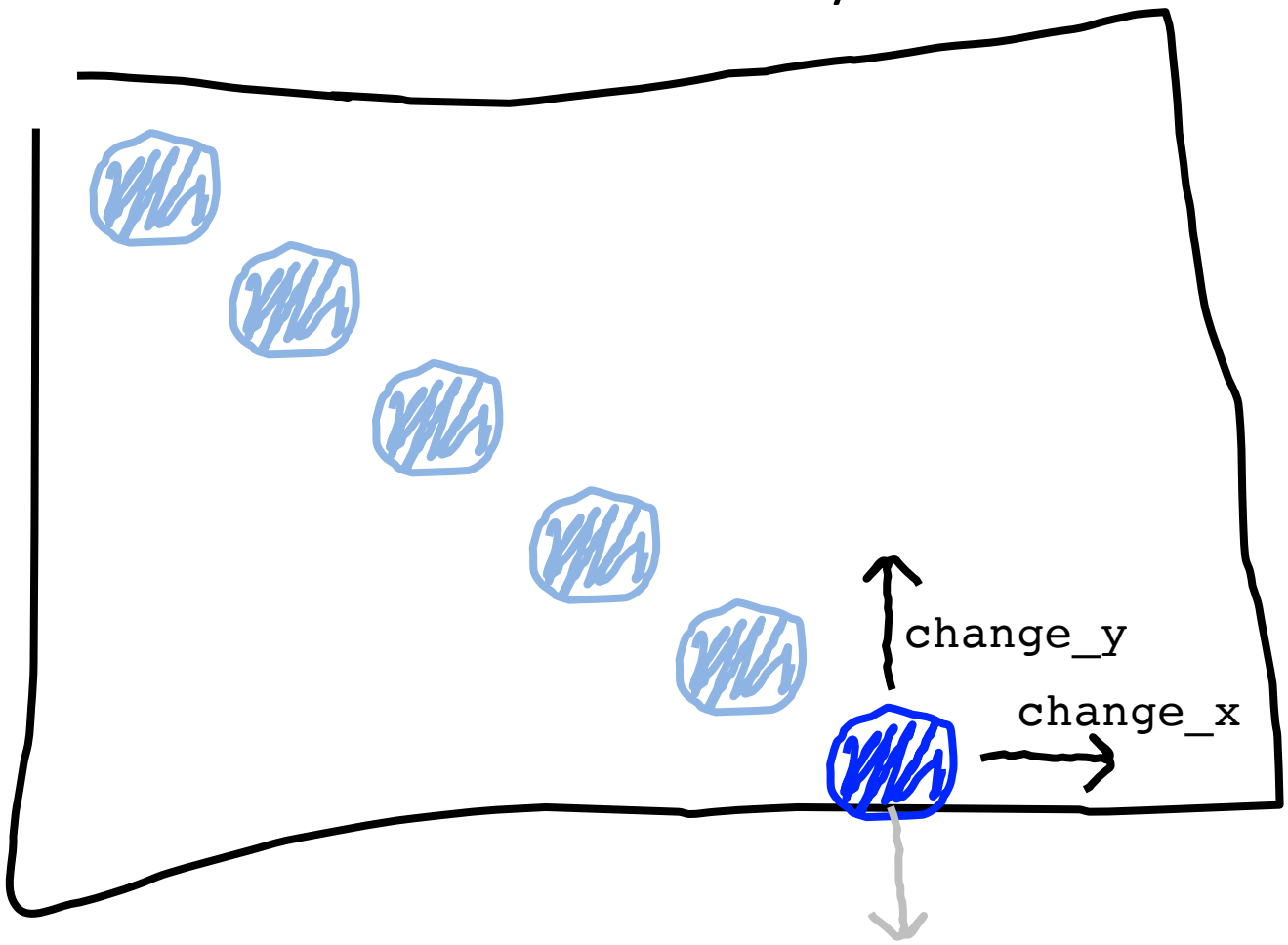
Bouncing Ball

We have this velocity

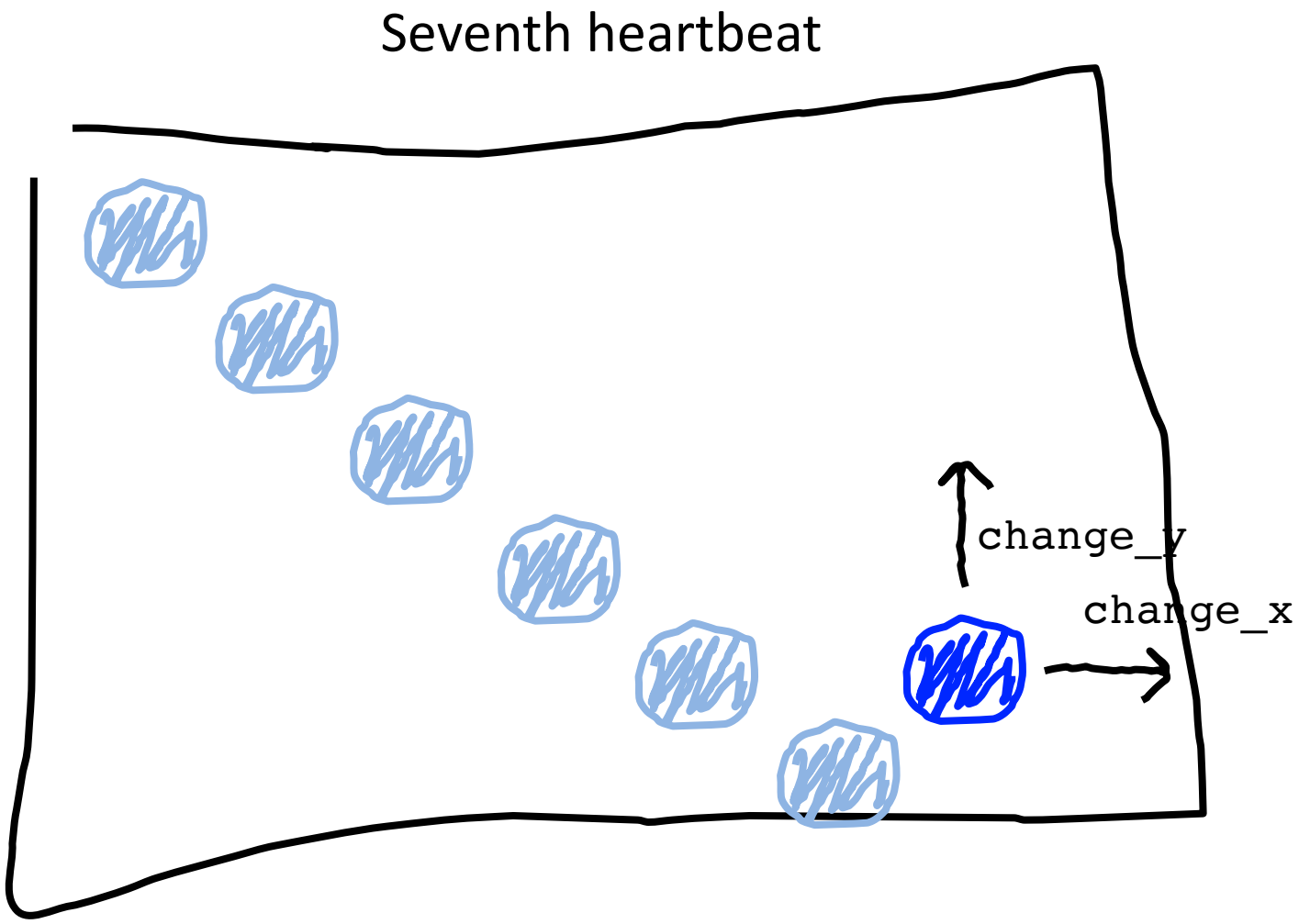


Bouncing Ball

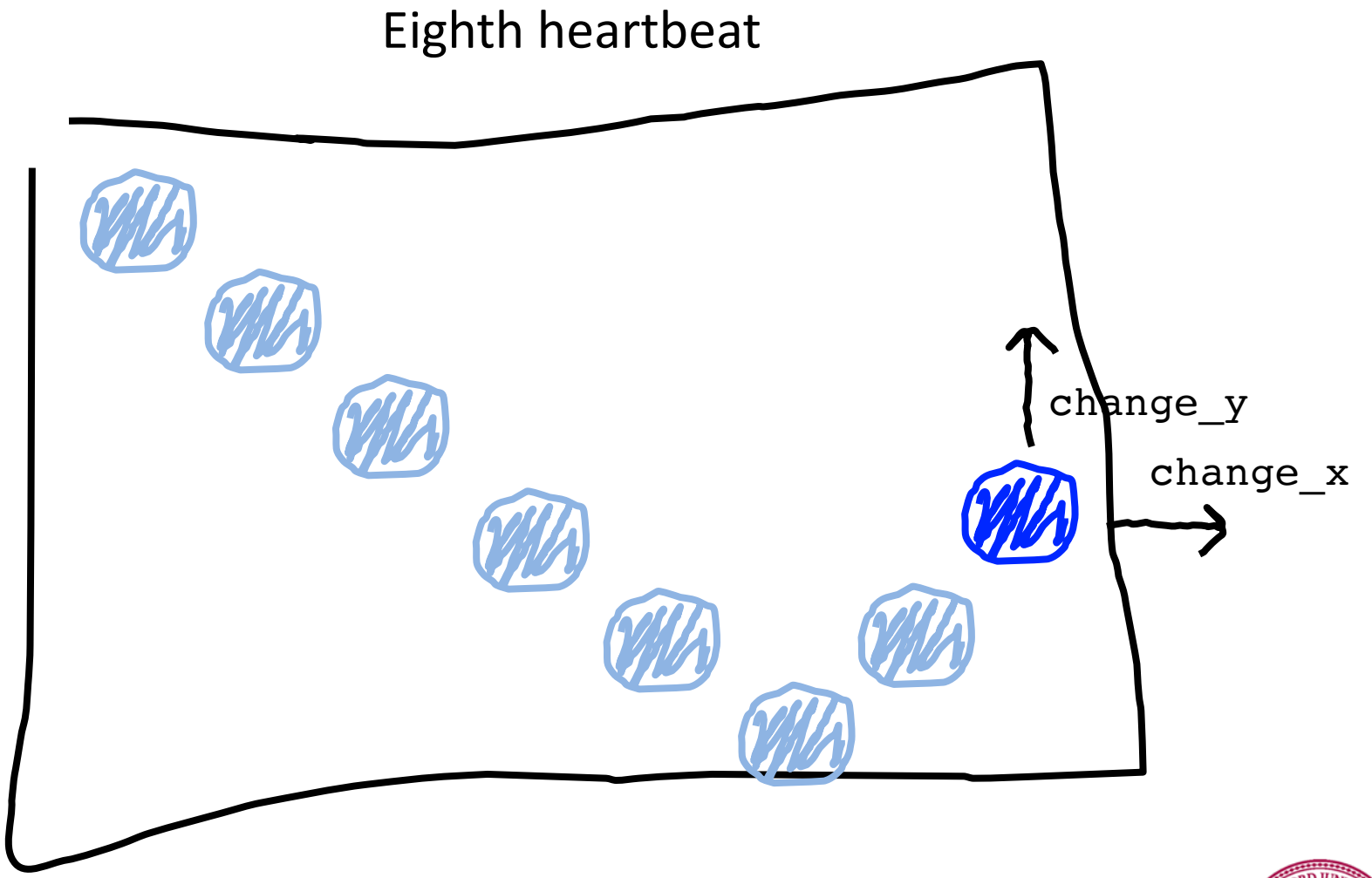
Our new velocity



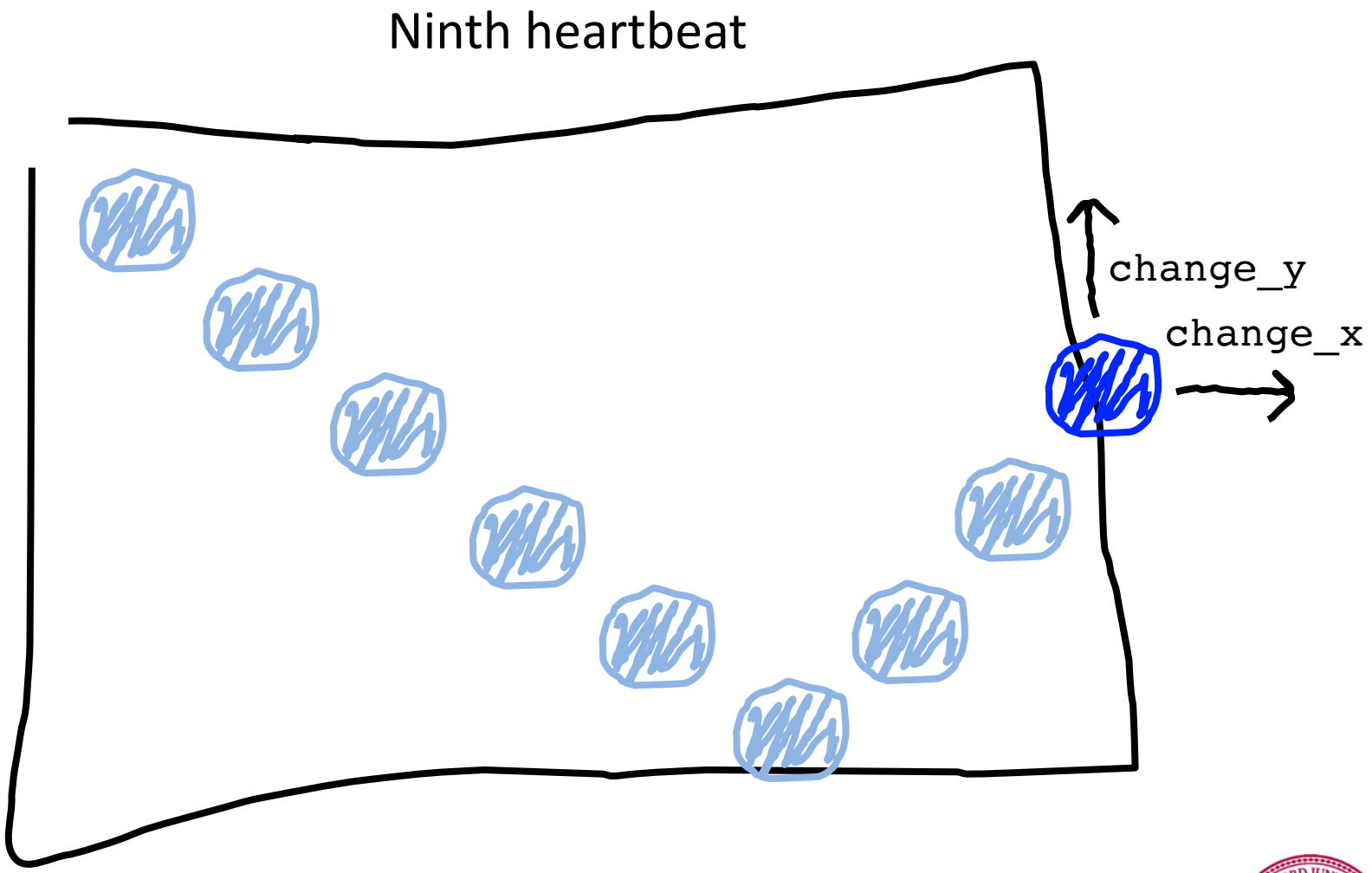
Bouncing Ball



Bouncing Ball

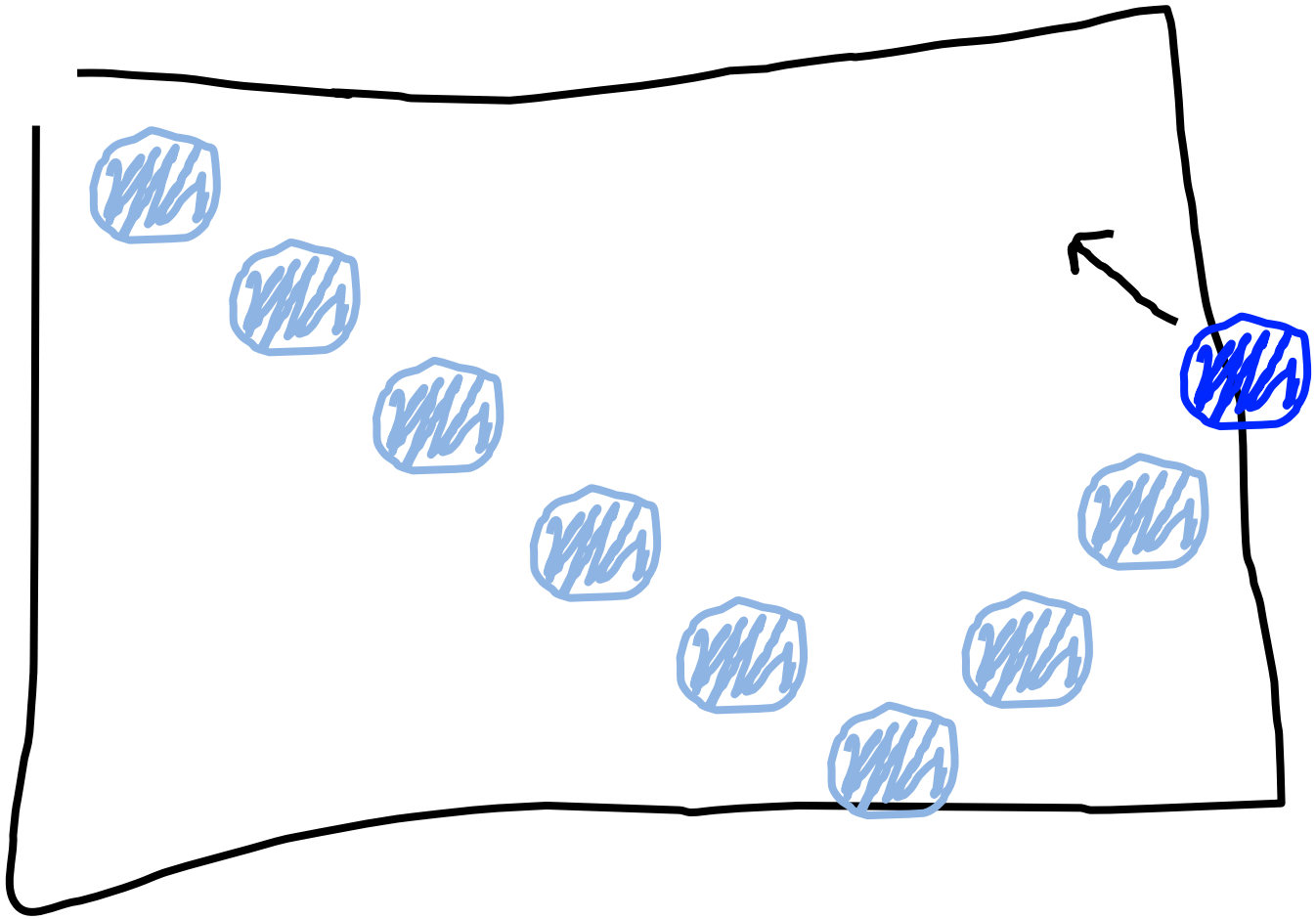


Bouncing Ball

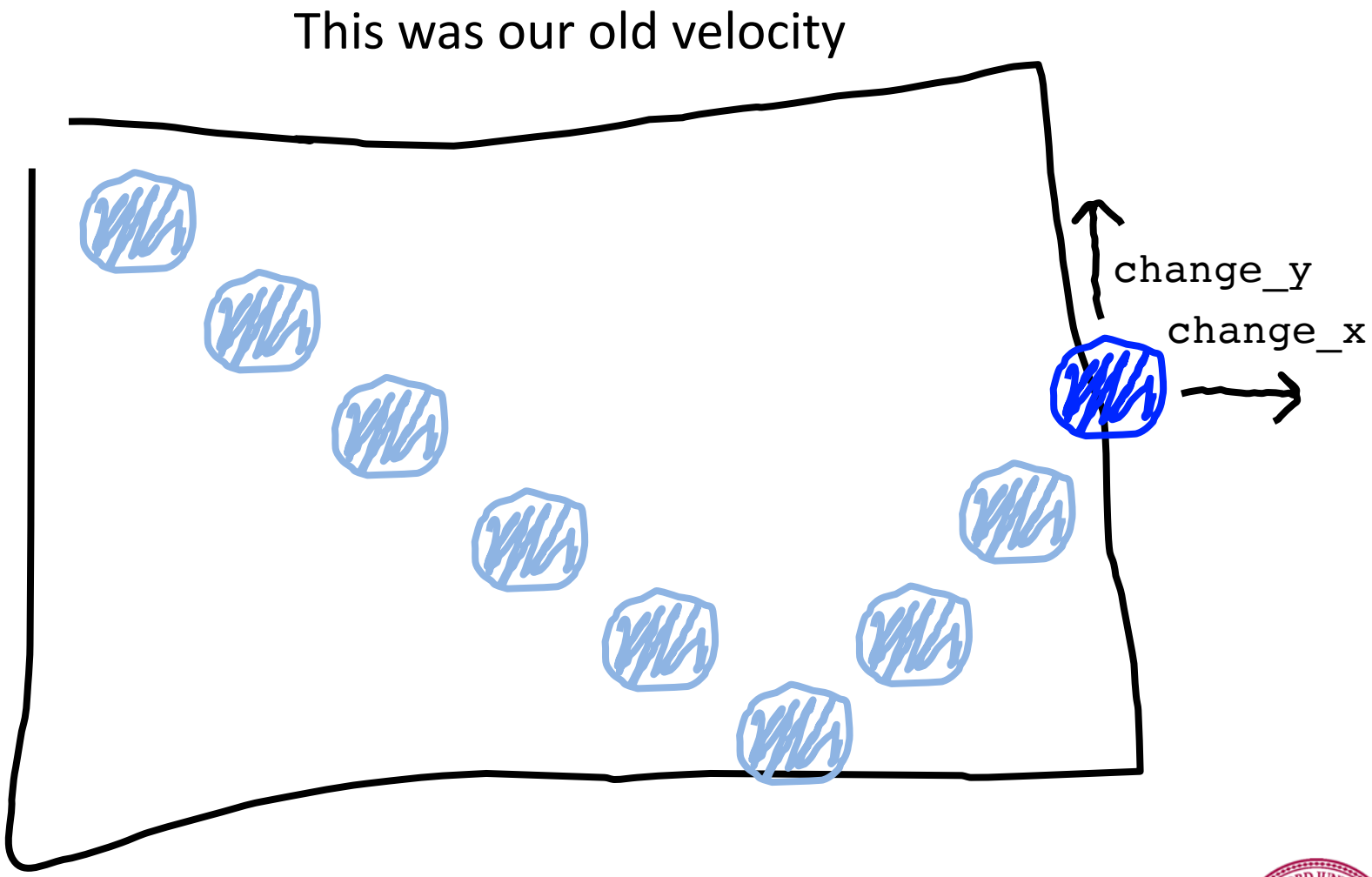


Bouncing Ball

We want this!

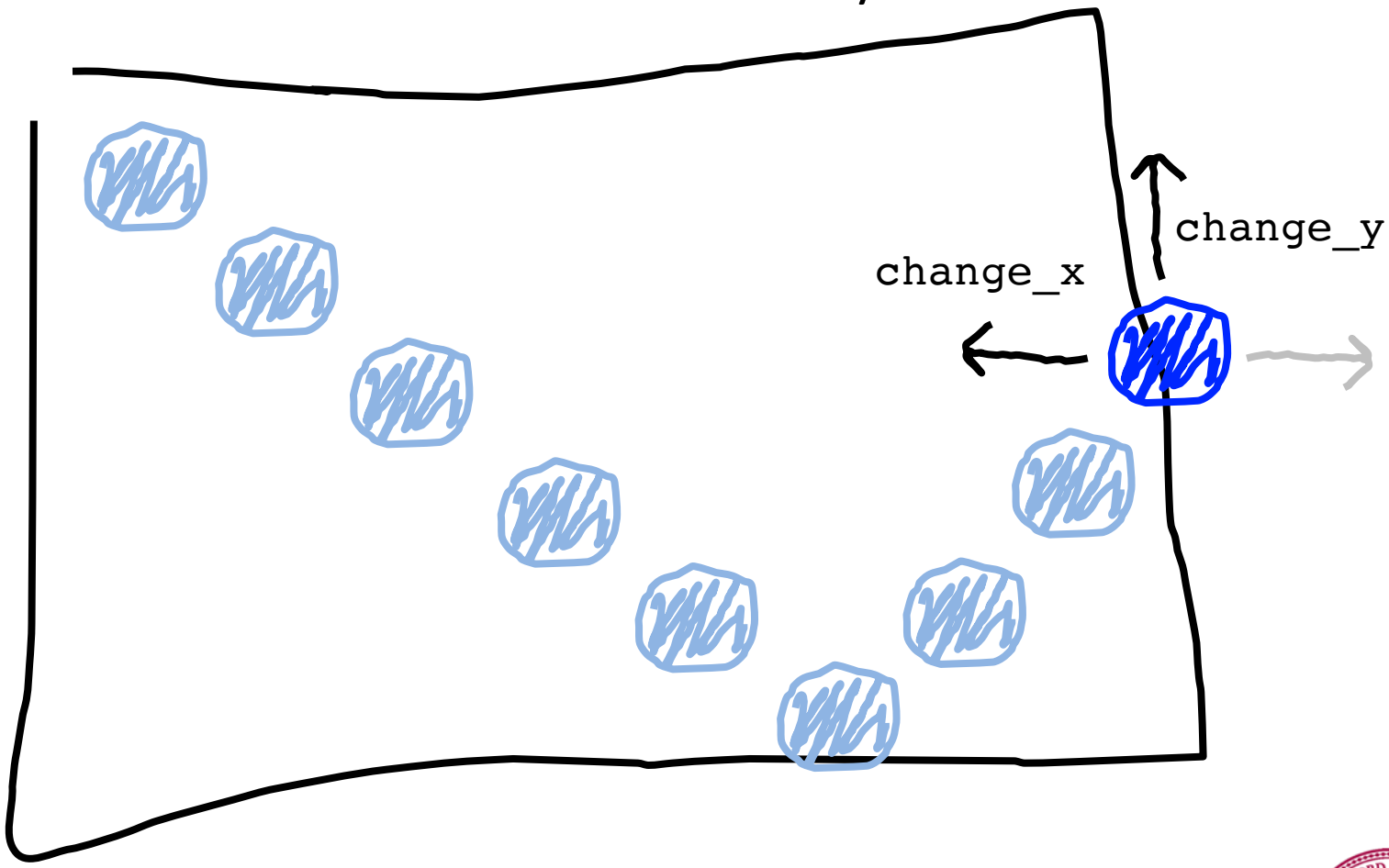


Bouncing Ball



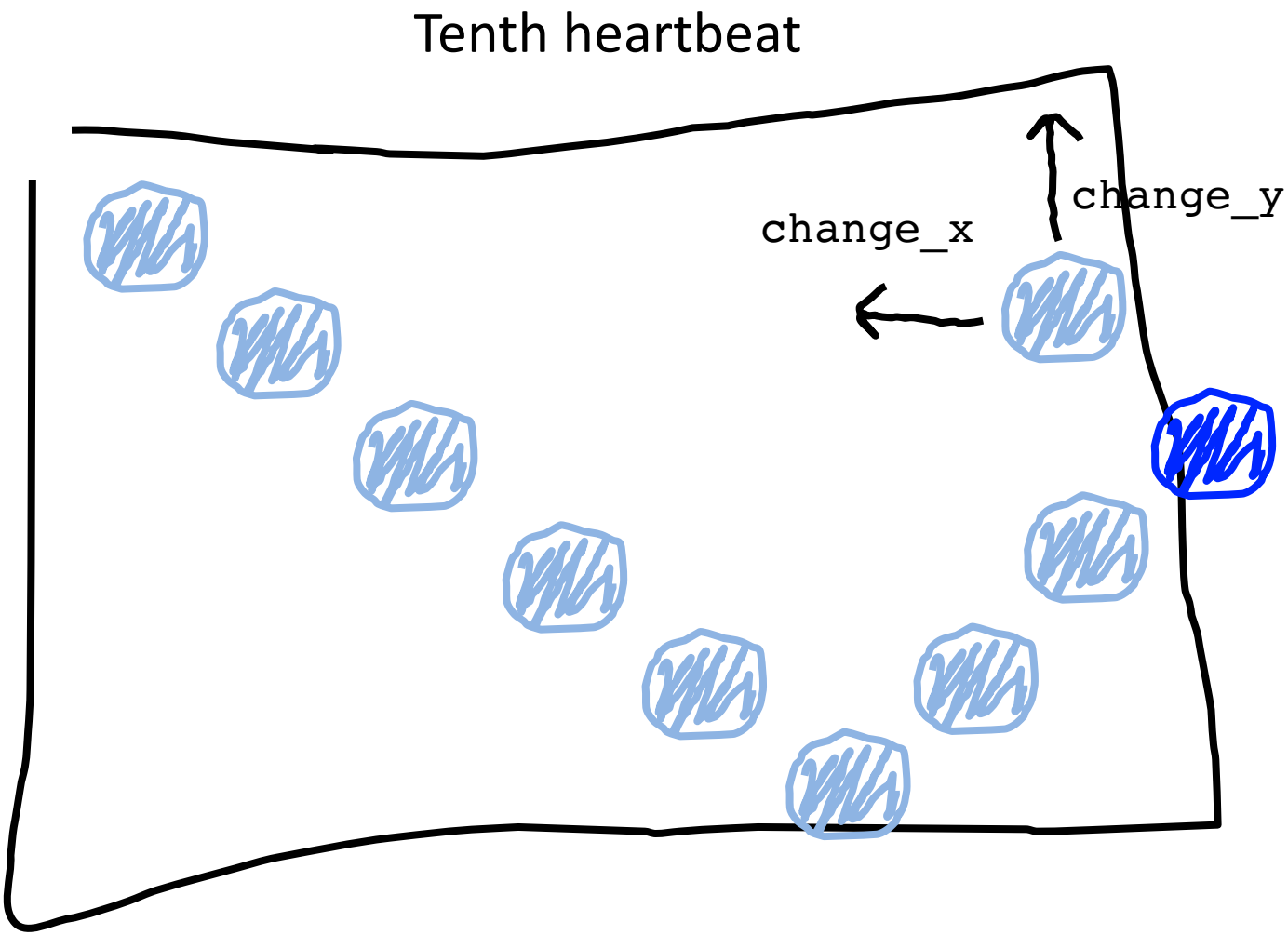
Bouncing Ball

This is our new velocity



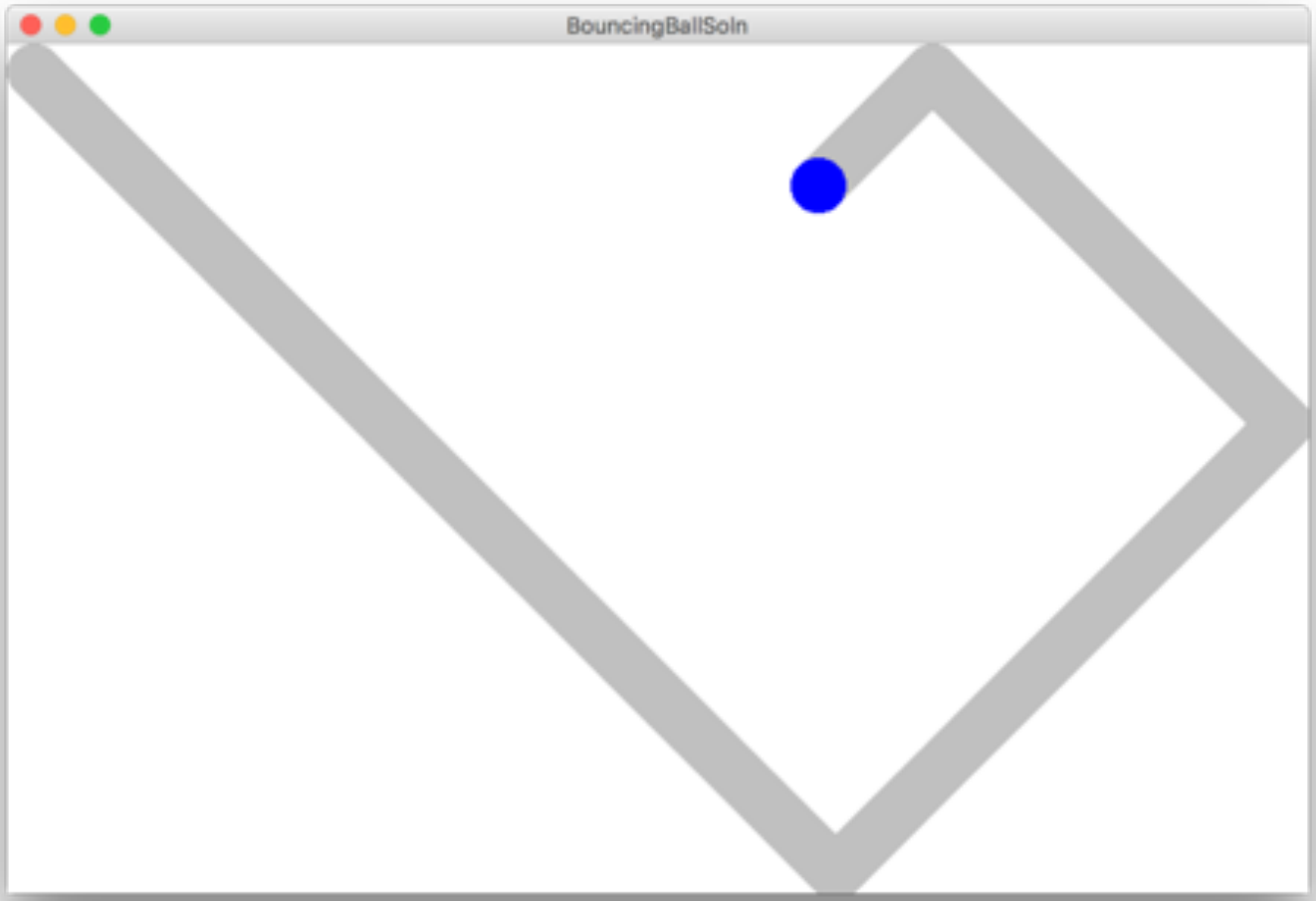
When reflecting horizontally: `change_x = -change_x`

Bouncing Ball



When reflecting horizontally: `change_x = -change_x`

Bouncing Ball



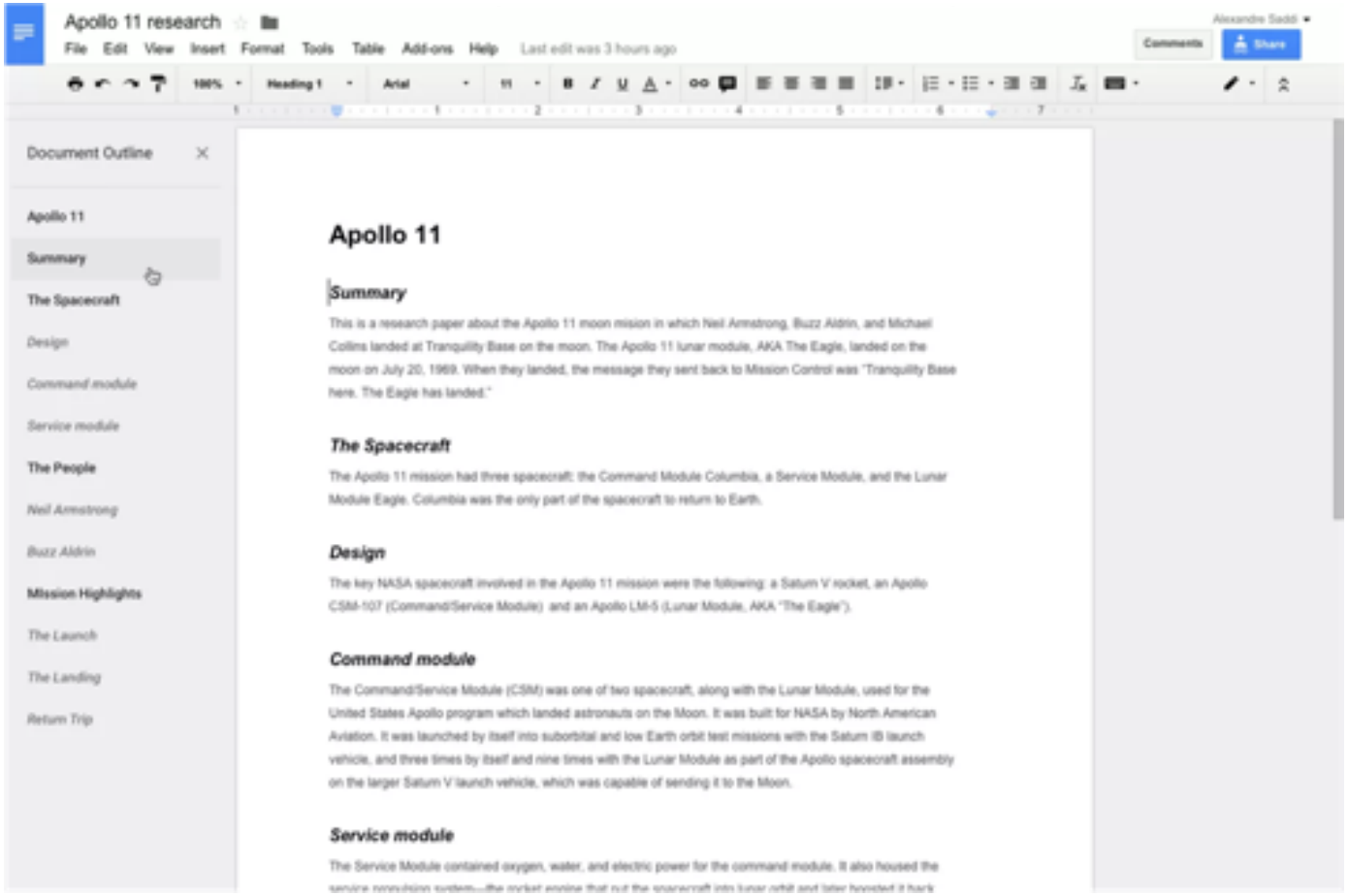
Hold up!

```
def make_ball(canvas):
```

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

*Large variables (objects) are stored using a reference which is like a **URL**. The URL gets copied when you pass the variable*

How do you share google docs?



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

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

def make_ball(canvas):
    canvas.create_rectangle( … , fill='blue')
```

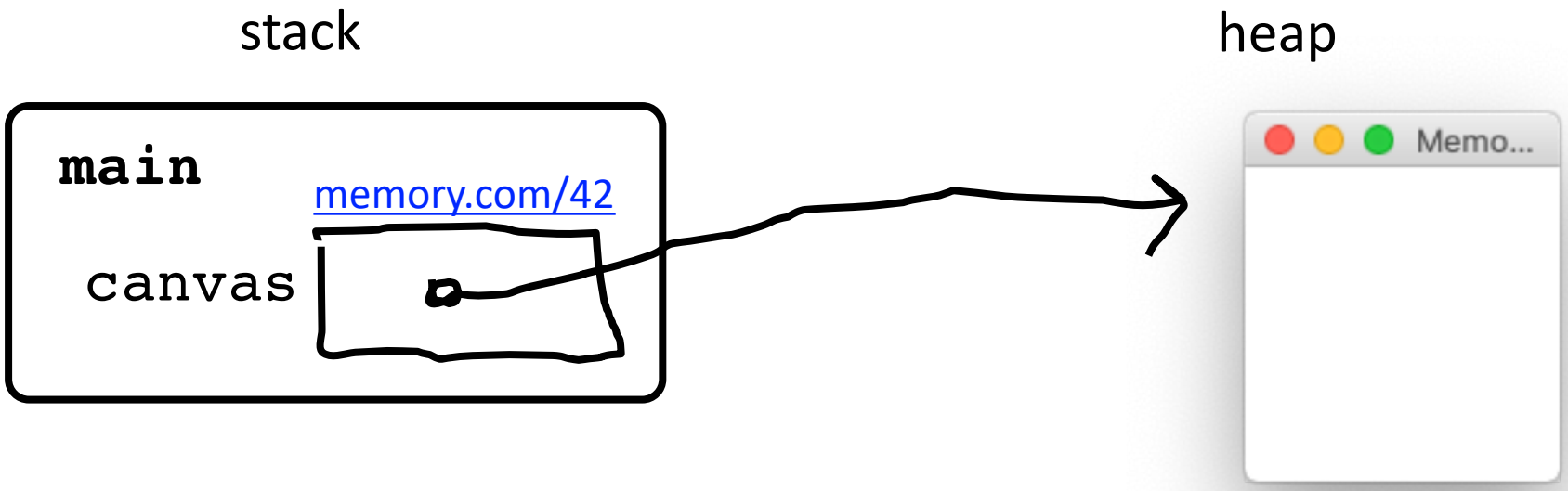
---

stack

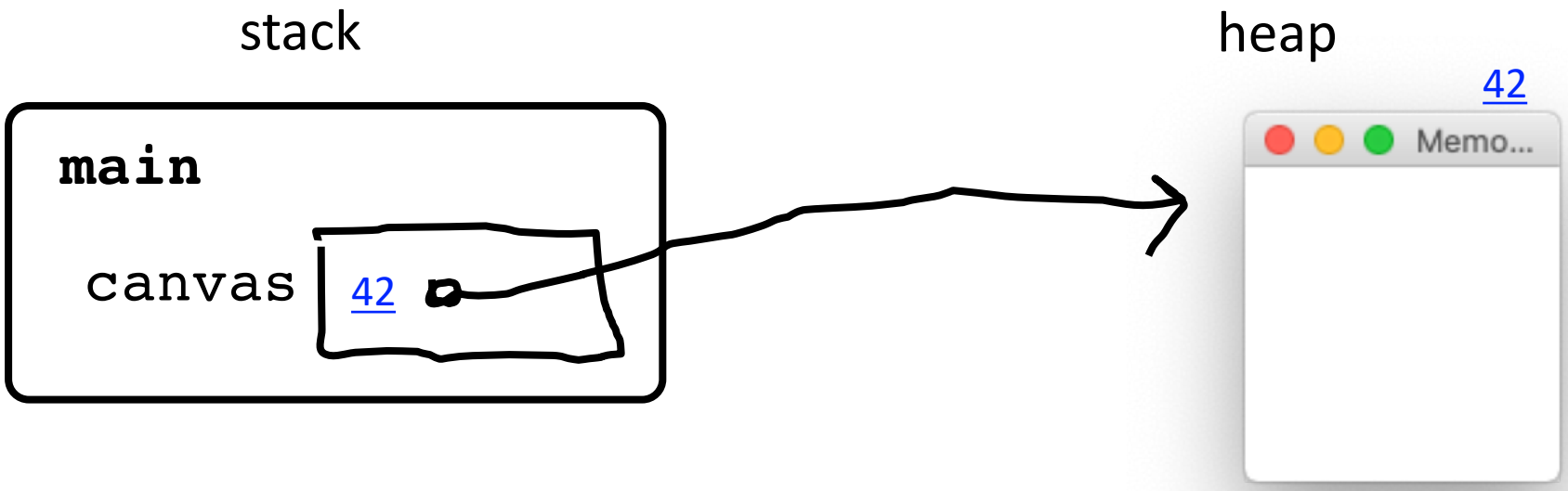
heap

main

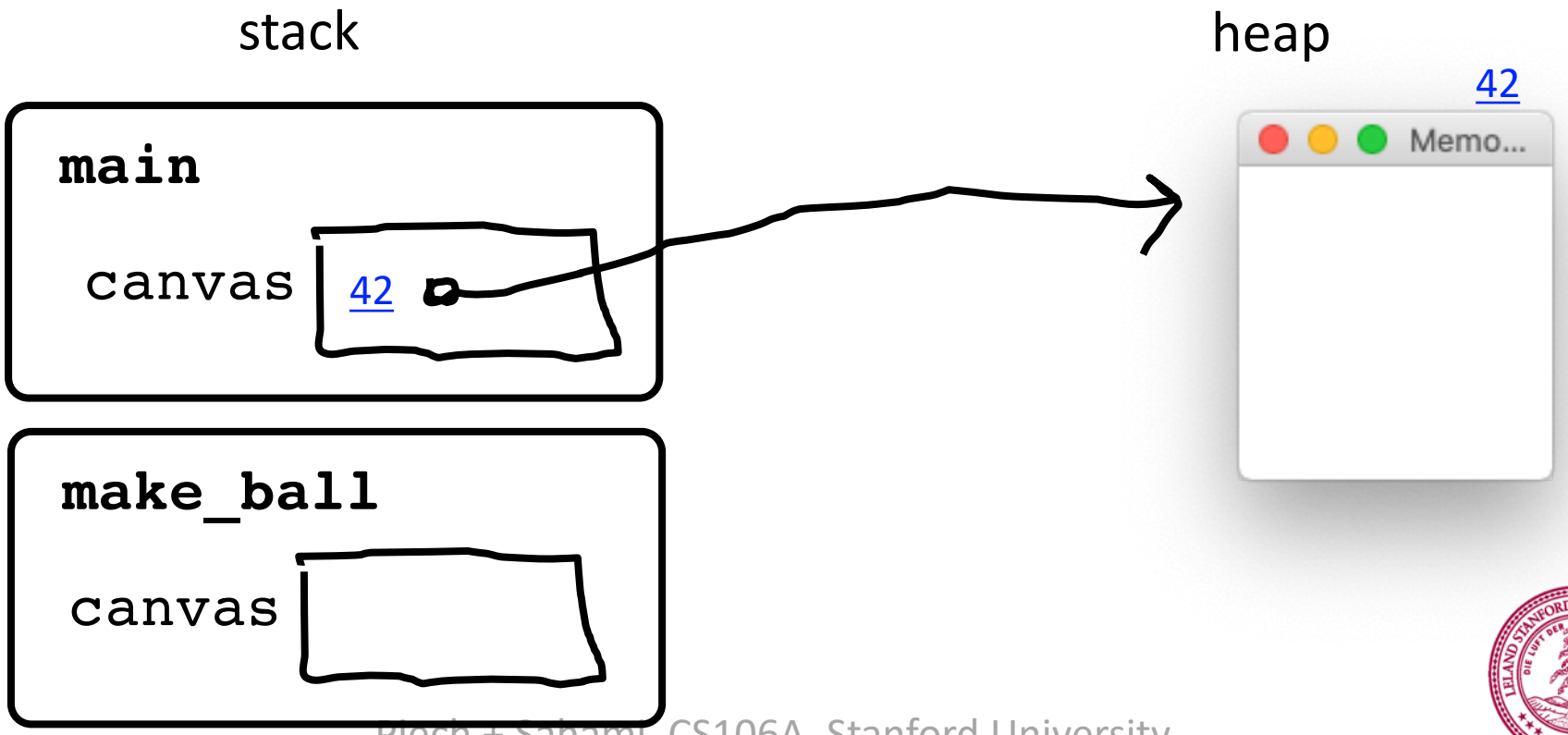
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_rectangle( … , fill='blue')
```



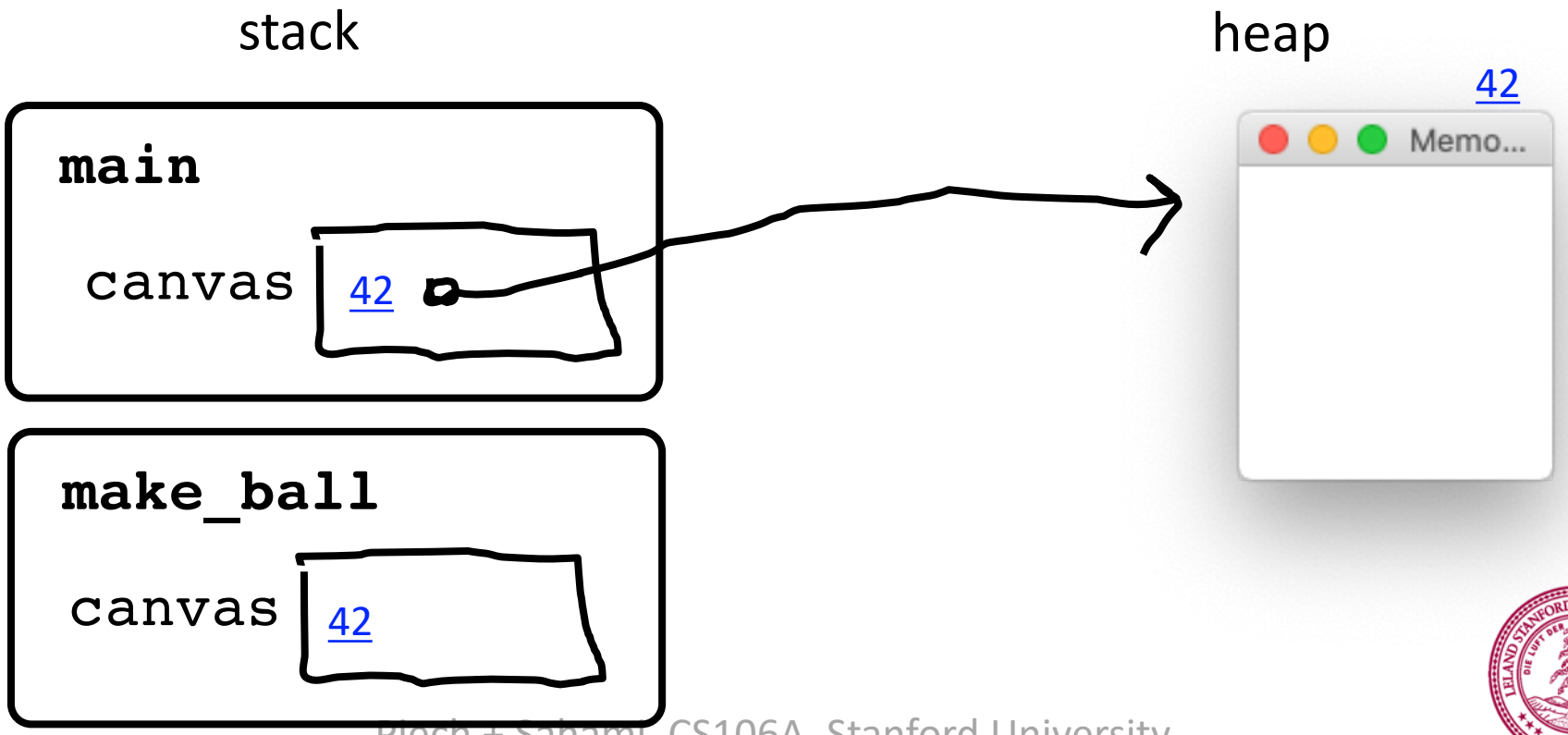
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



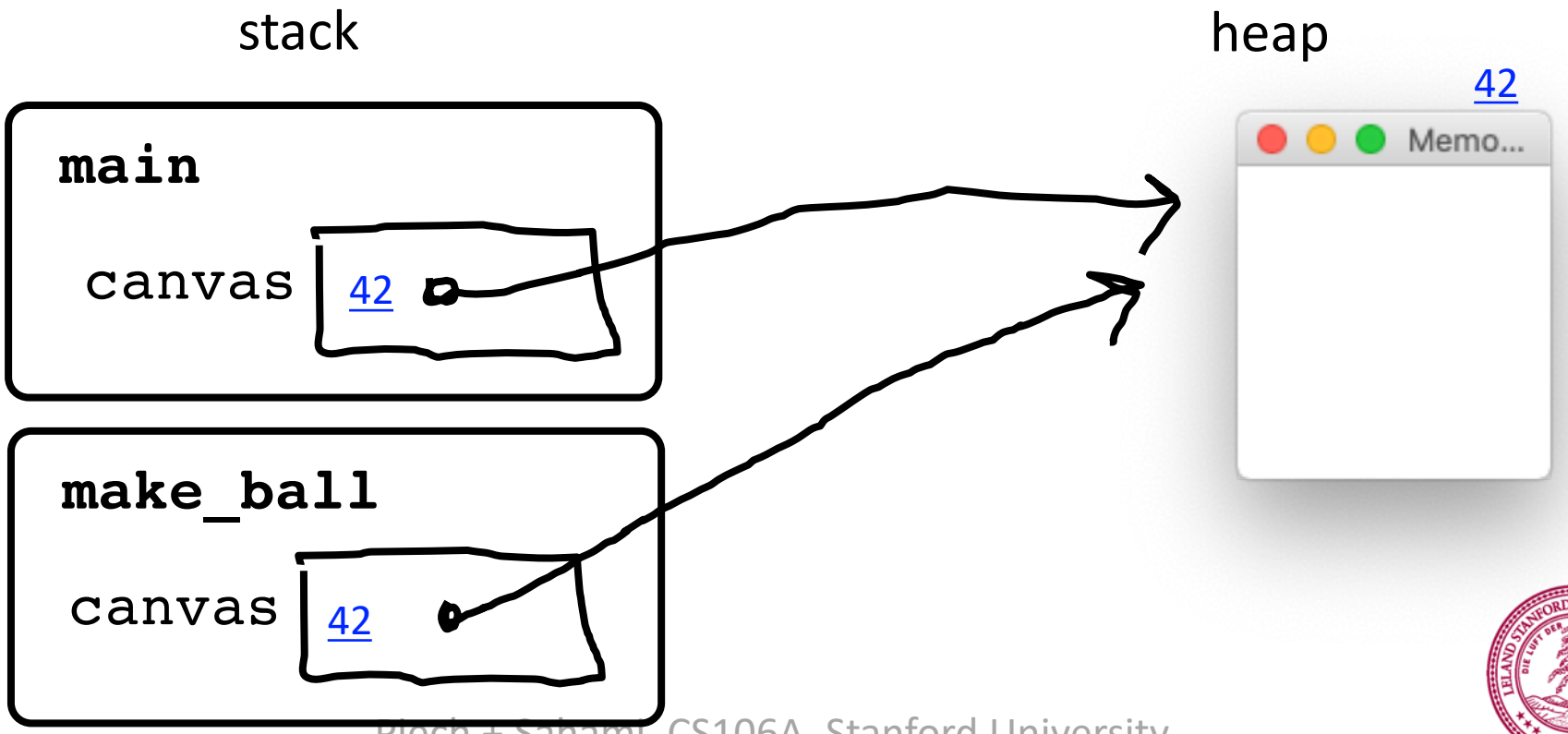
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



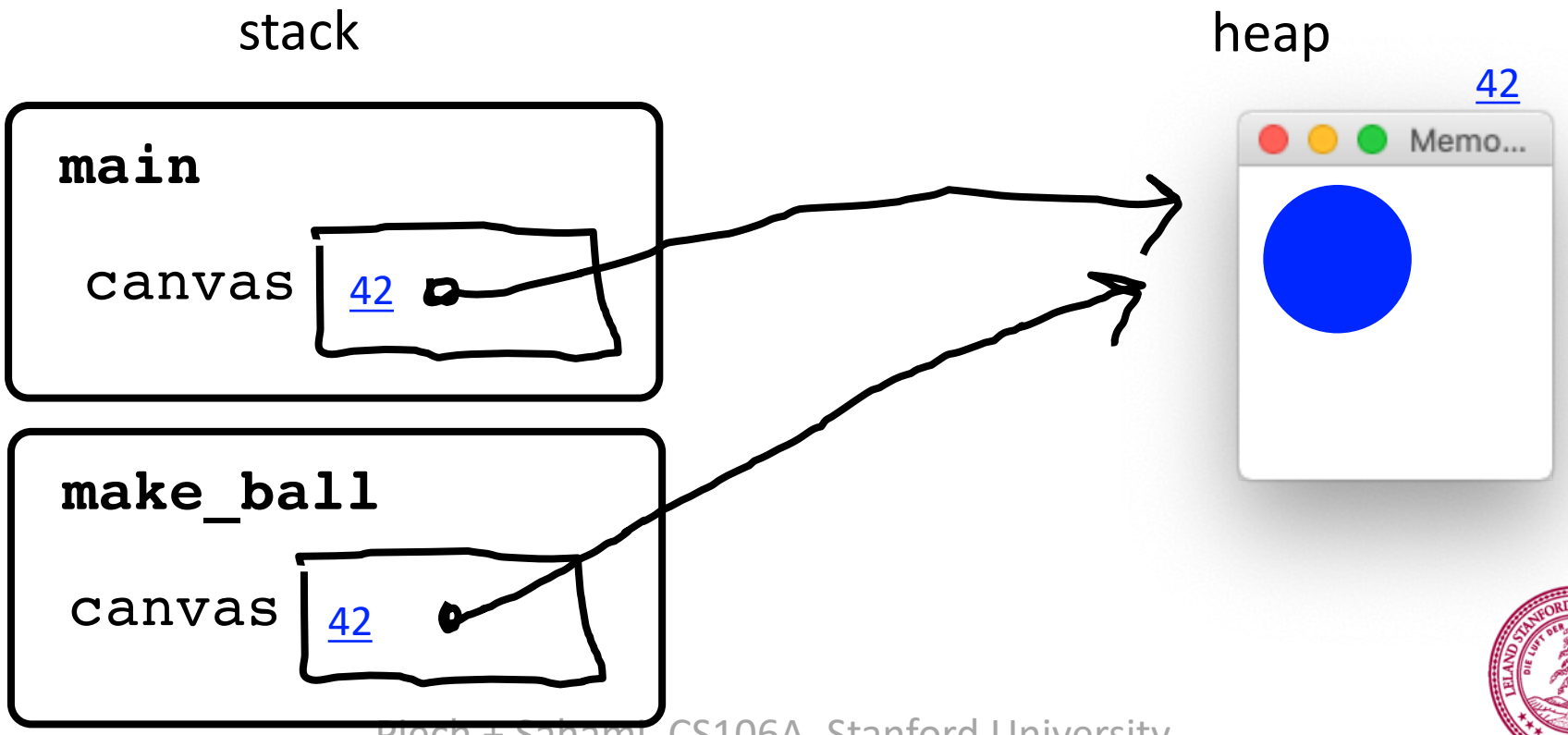
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



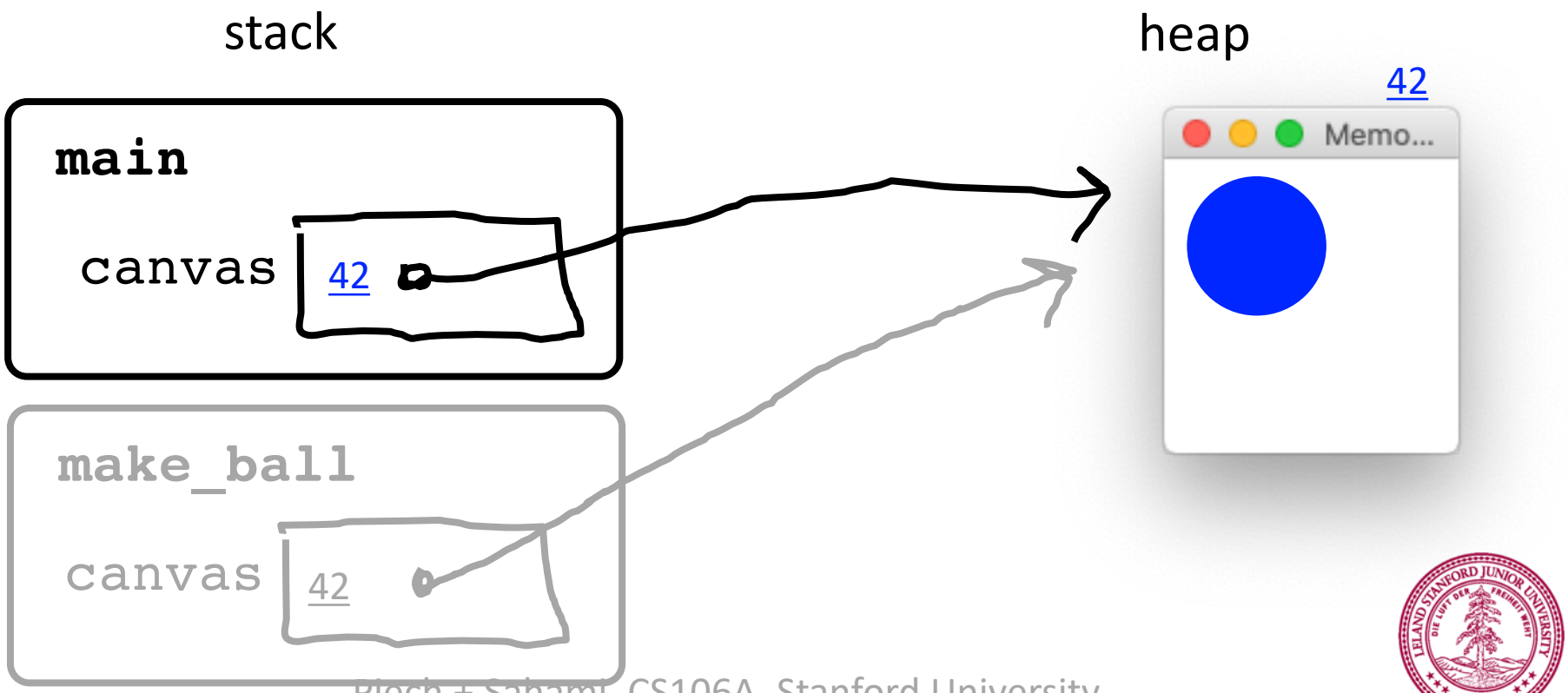
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



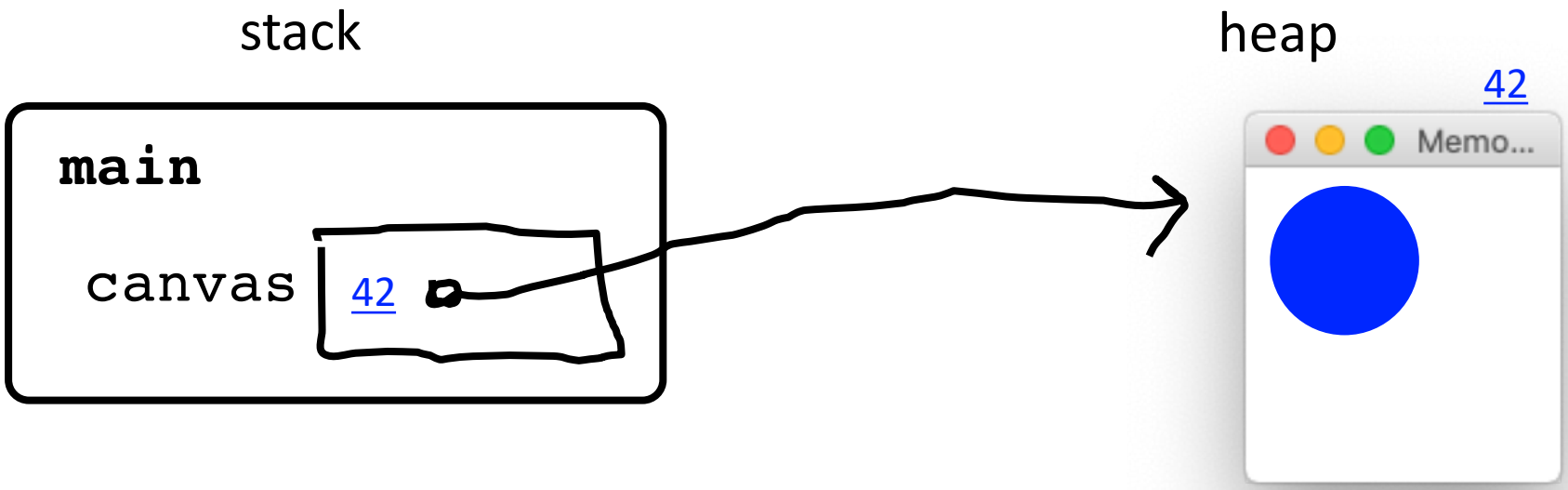
```
def main():
    canvas = make_canvas(…)
    make_ball(canvas)

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



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

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



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

def make_ball(canvas):
    canvas.create_oval( … , fill='blue')
```



Some variables are stored with references

(which are like memory URLs)

When passed as parameters

Variables that act like their **value is copied**

```
boolean
integer
float
string
```

Variables that act like their **URL is copied**

```
canvas
pixel
SimpleImage
list
```

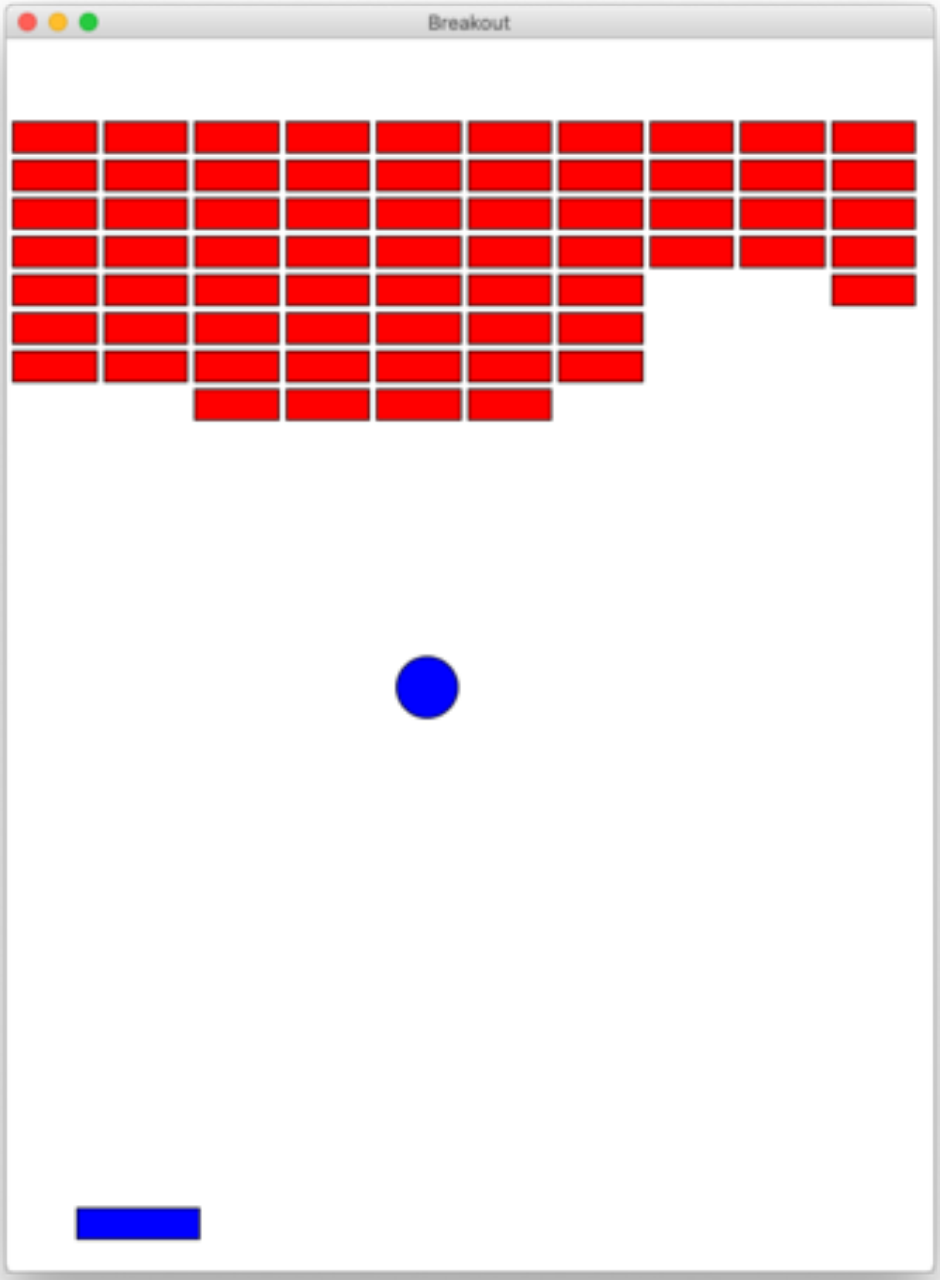
Learning Goals

1. Feel more confident writing methods
2. Write animated programs

Special Graphics Functions

```
# get the x location of the mouse
mouse_x = canvas.winfo_pointerx()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)

# move shape by a given change_x and change_y
canvas.move(shape, change_x, change_y)

# get the coordinates of a shape
coord_list = canvas.coords(shape)

# return a list of elements in a rectangle area
results = canvas.find_overlapping(x1, y1, x2, y2)

# you can change a shapes color too
canvas.itemconfig(shape, fill=new_color, outline=...)
```

Come back on Monday to learn about lists!


Breakout