

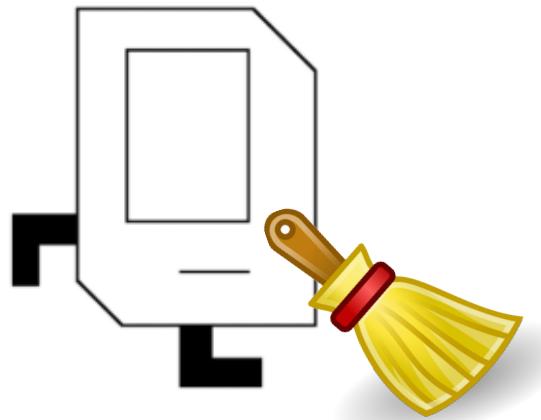


Animation

Chris Gregg

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

Housekeeping



- Reminder: Diagnostic is on **Thursday, July 9th**
 - Takes place during class time
 - Download link for exam .json file (will be available July 9th at 10:30am, PDT):
 - <https://web.stanford.edu/class/cs106a/cgi-bin/diagnostic-bluebook/get-password.html>
 - Download the exam data and get started immediately
 - The exam will be a full hour long, and Bluebook forces you to submit after one hour

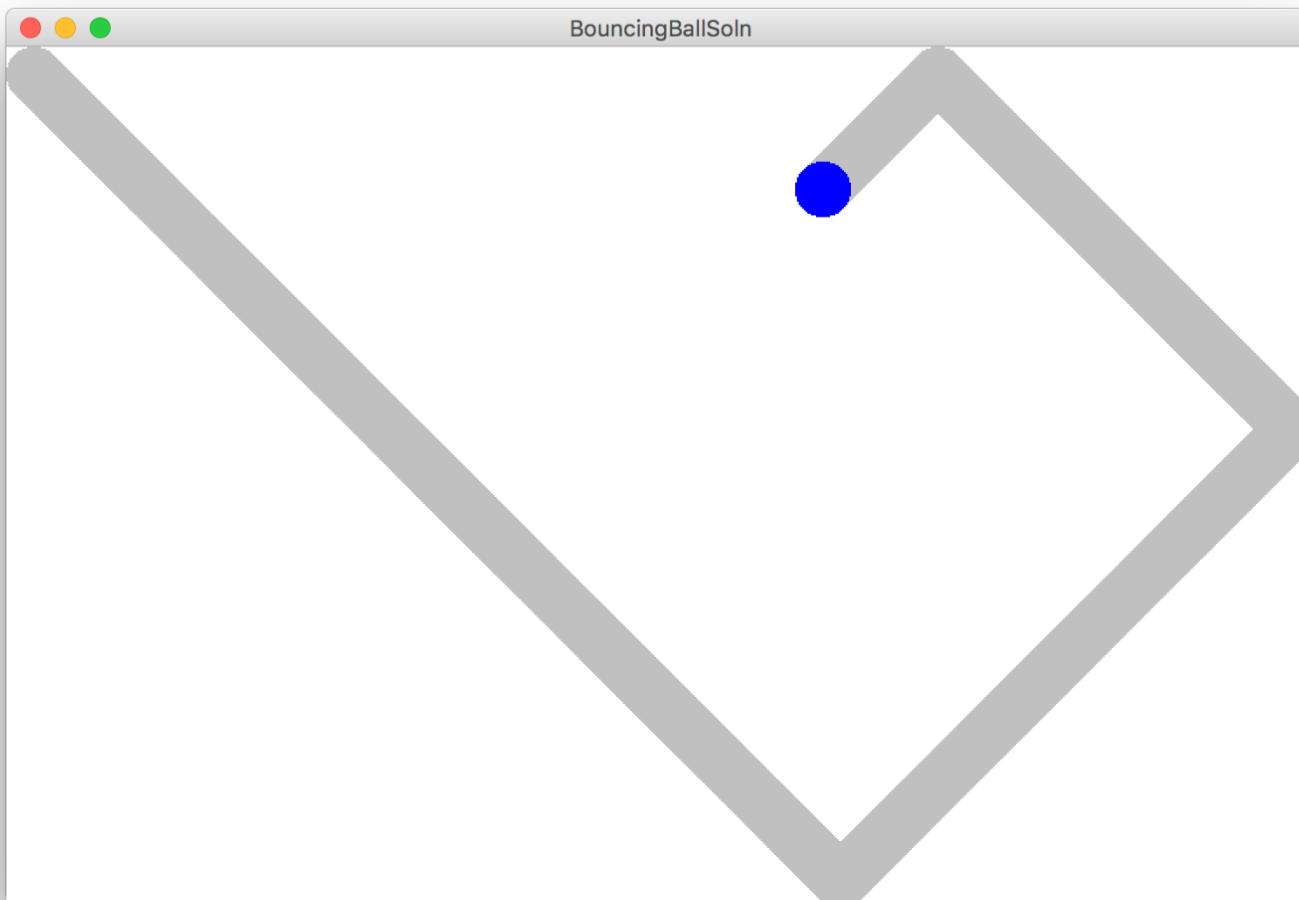


Learning Goals

1. Feel more confident debugging
2. Write animated programs



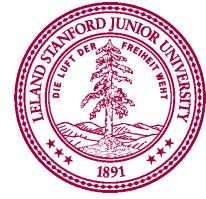
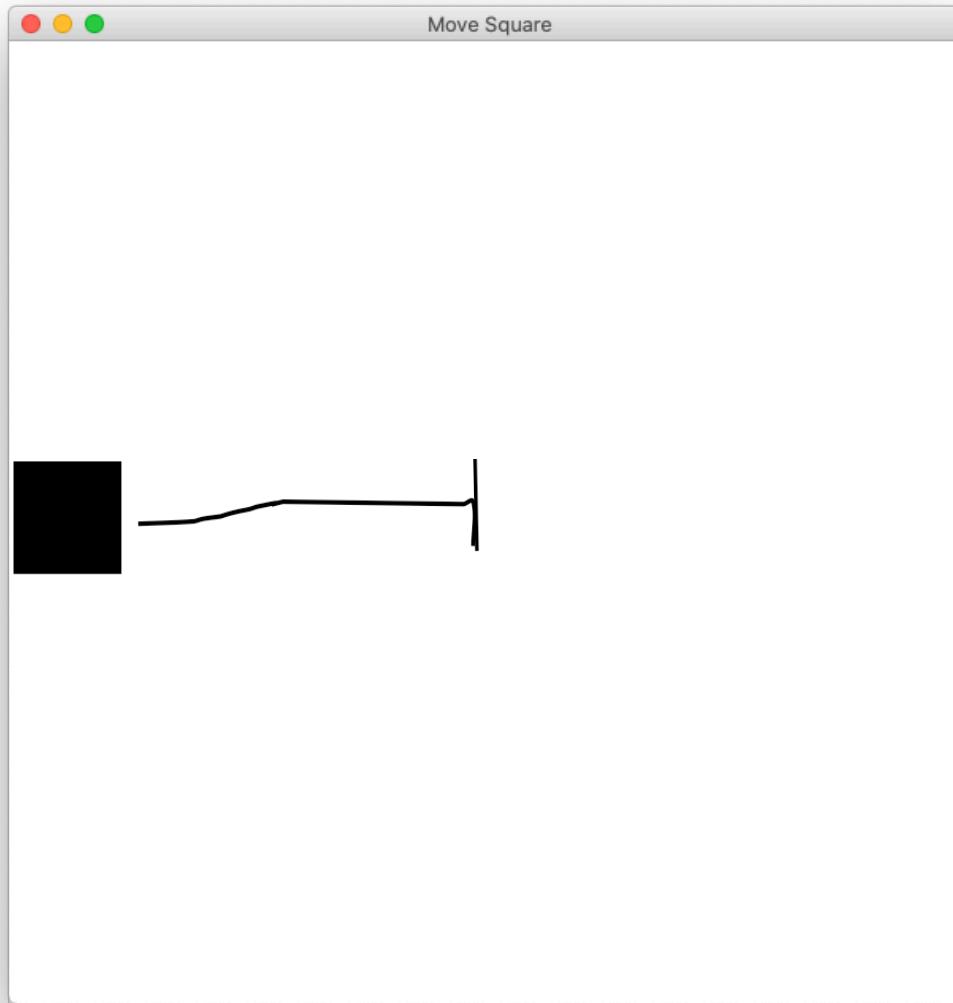
You will be able to write Bouncing Ball



Great foundation



Move to Center



In our last episode...

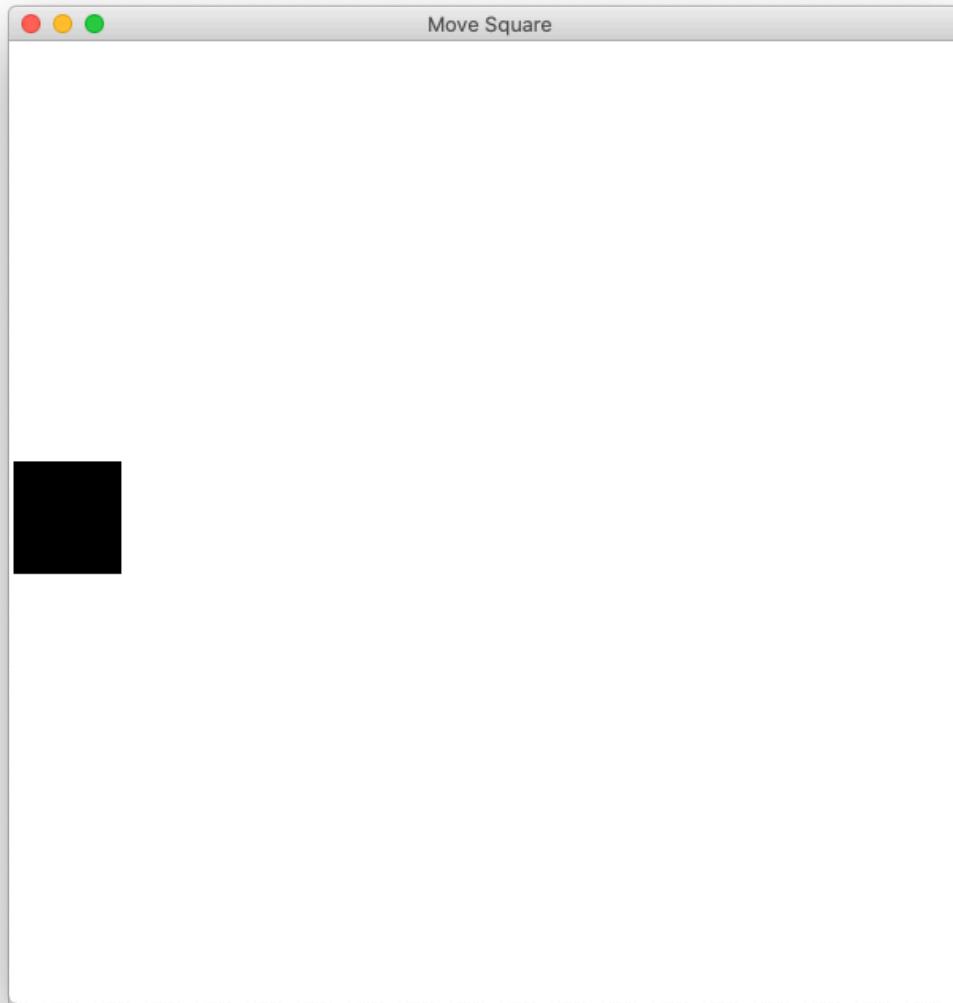
Graphics from tkinter (aka tk)

```
import tkinter
```

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



Add square



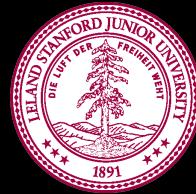
Graphics from tkinter (aka tk)

```
import tkinter
```

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



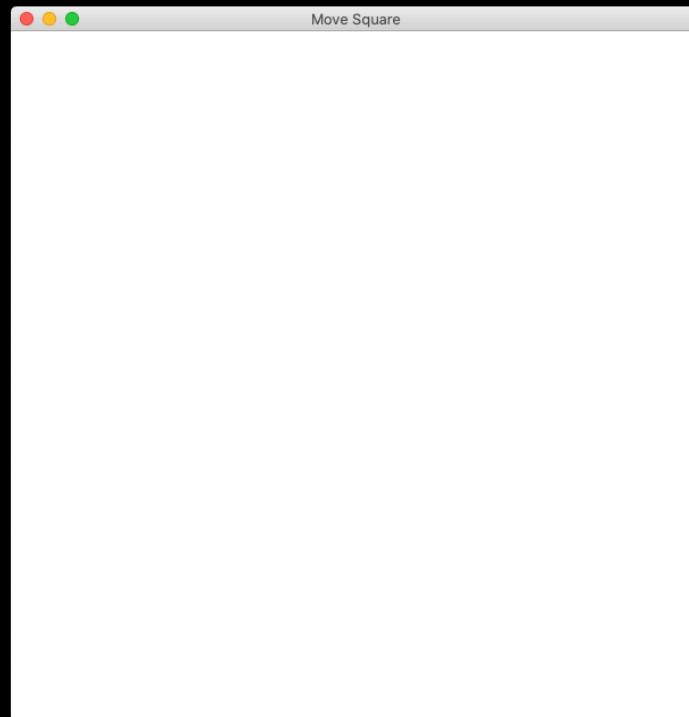
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



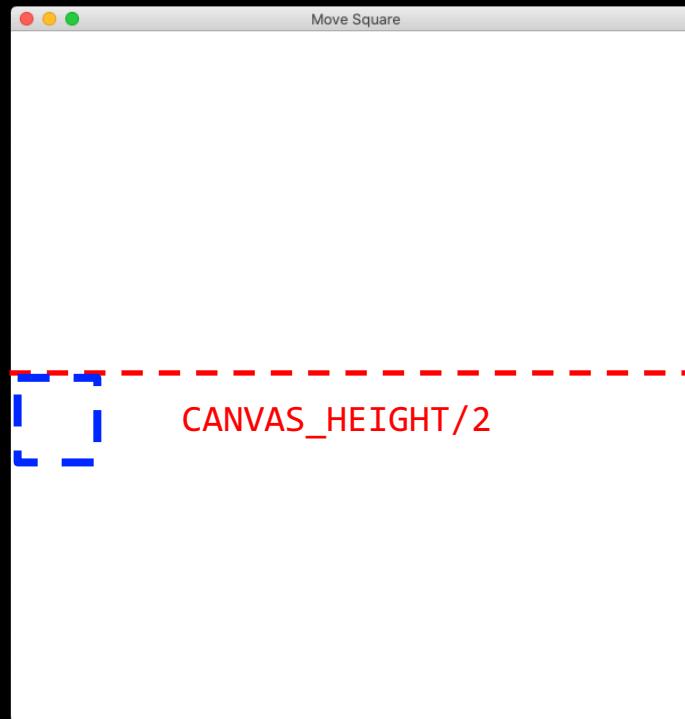
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



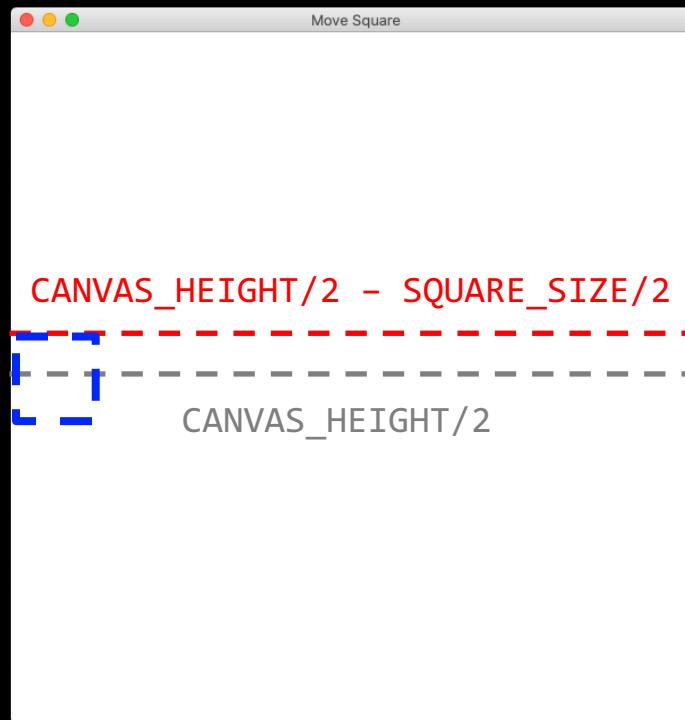
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



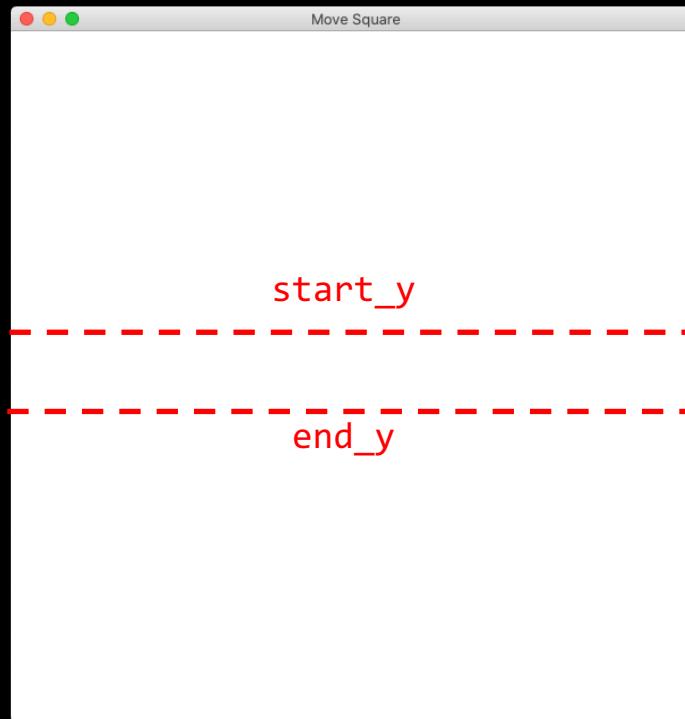
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



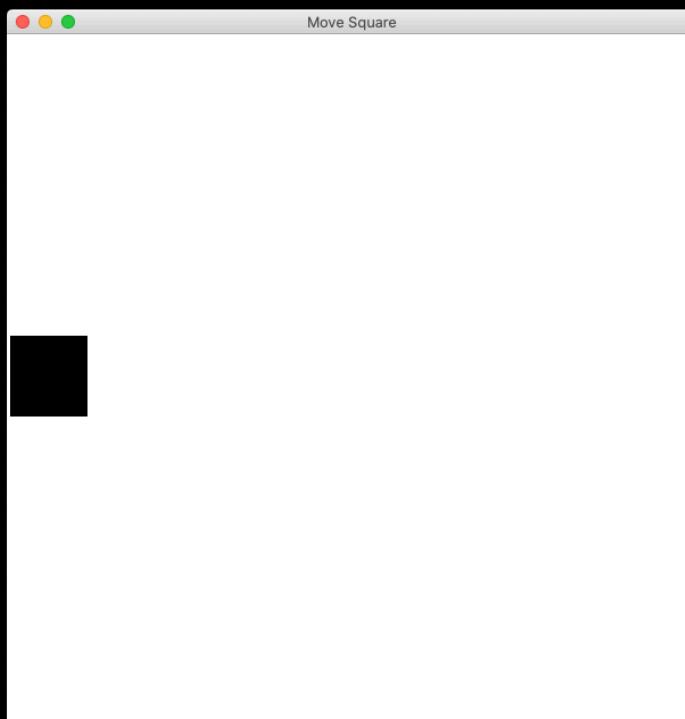
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



```
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()
```



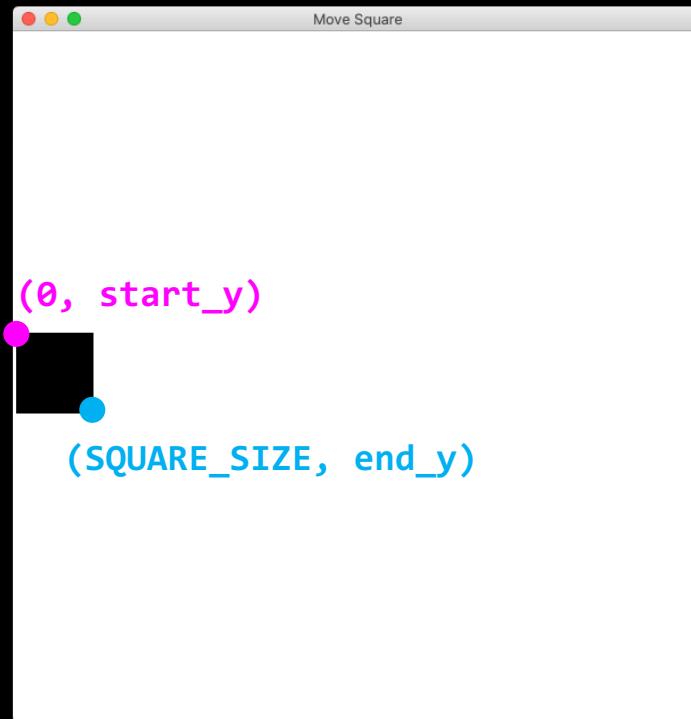
```
def main():
    canvas = make_canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
```



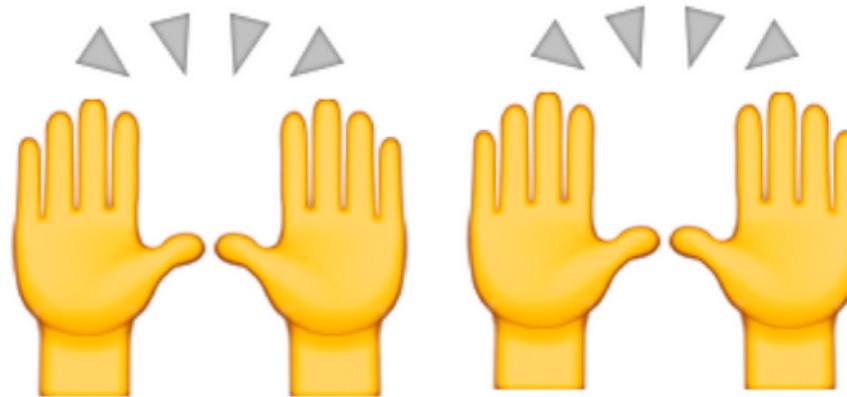
Some “heavy
duty” variables
allow you to call
functions on them



```
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!



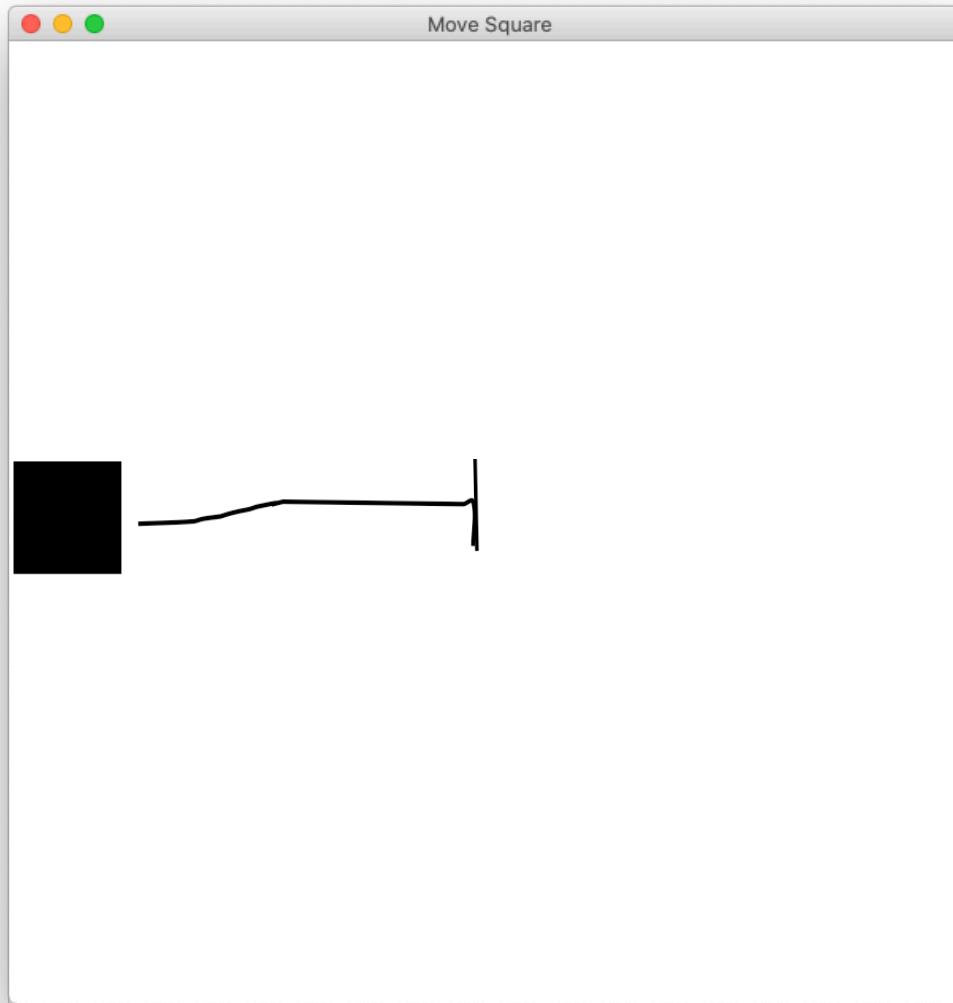
Woot!



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)
```



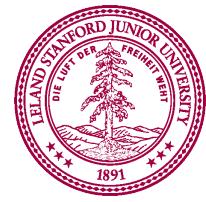
Animation Loop

```
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



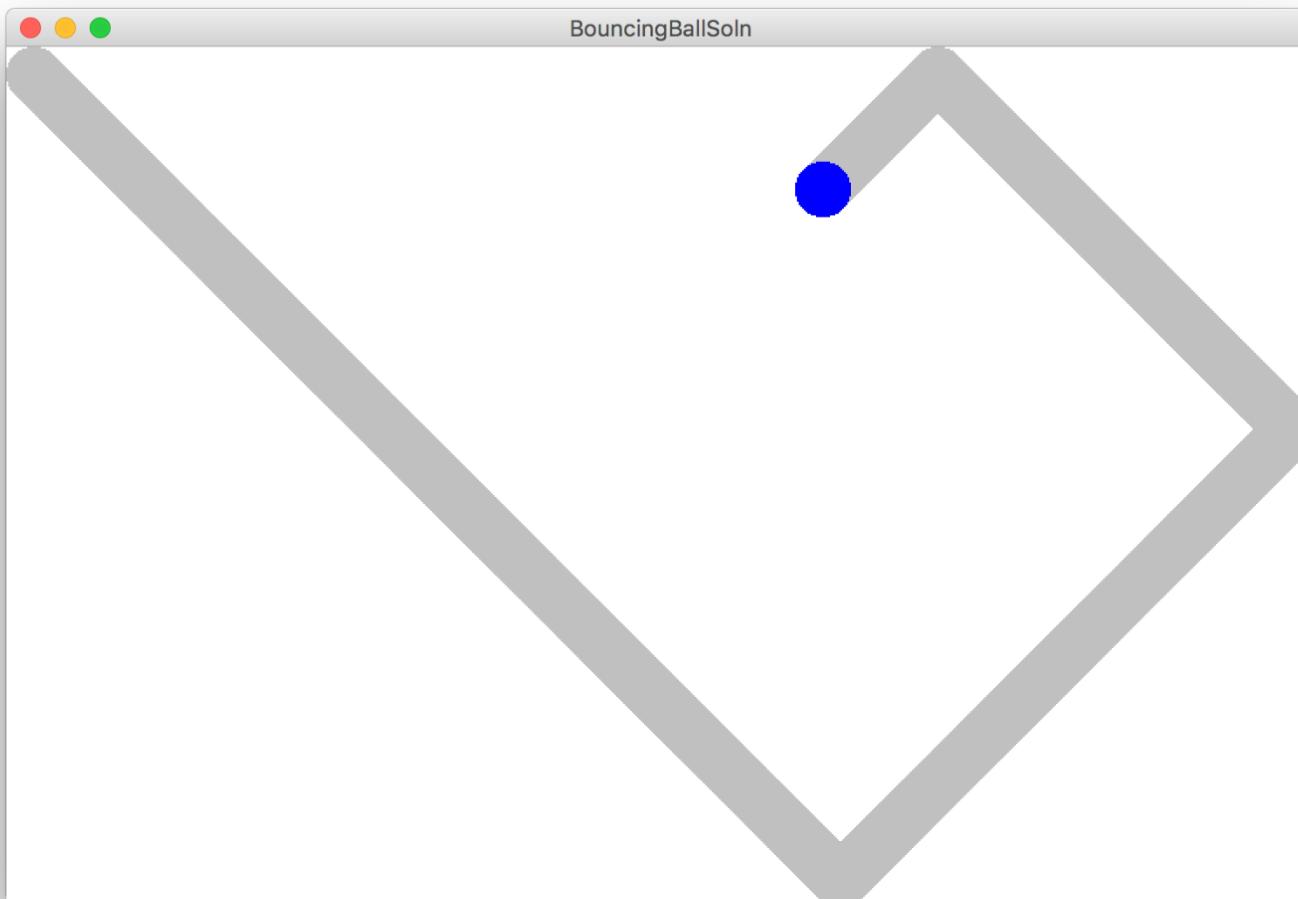
Move To Center

```
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)
```

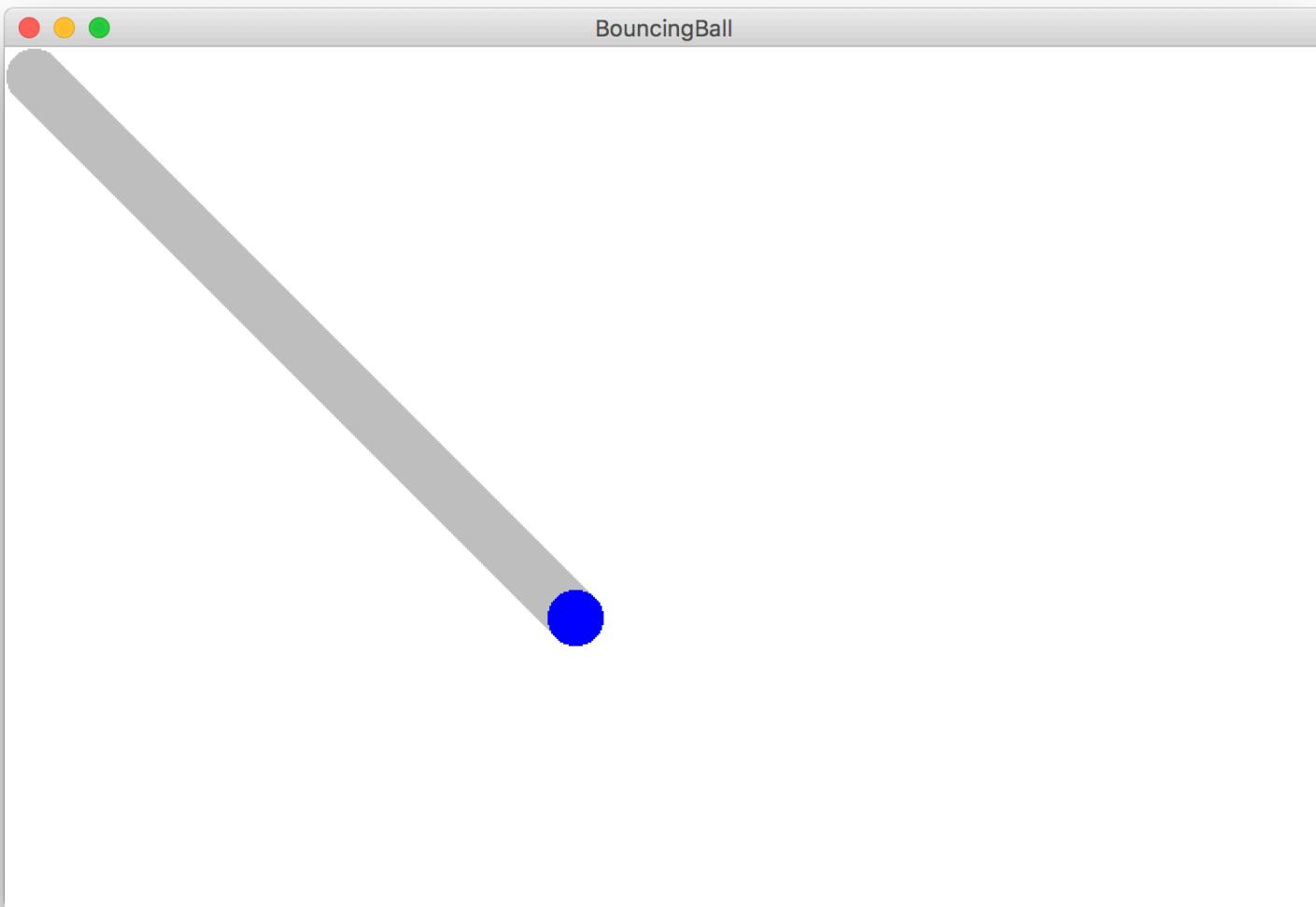


We are ready...

Bouncing Ball

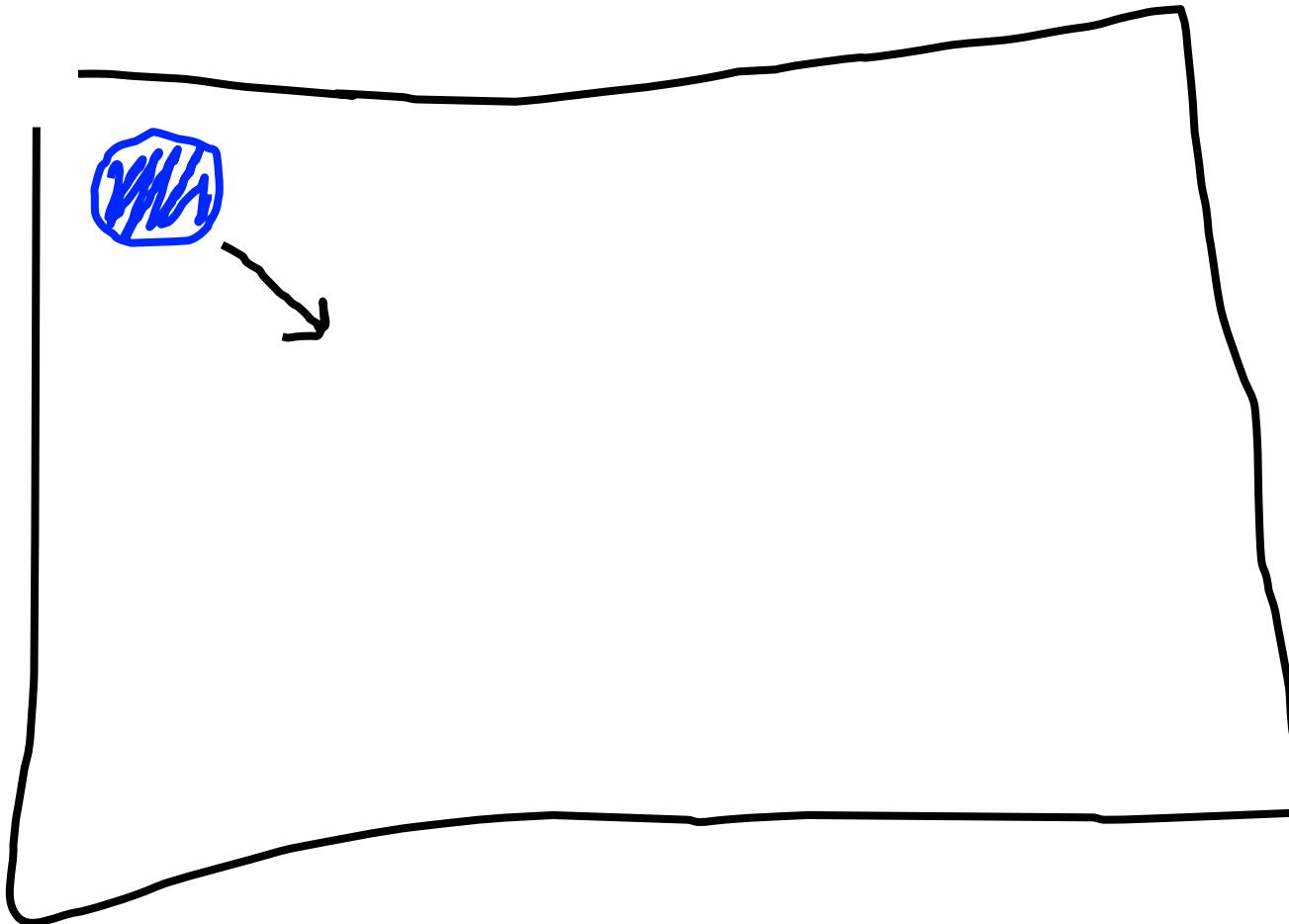


Milestone #1



Bouncing Ball

First heartbeat

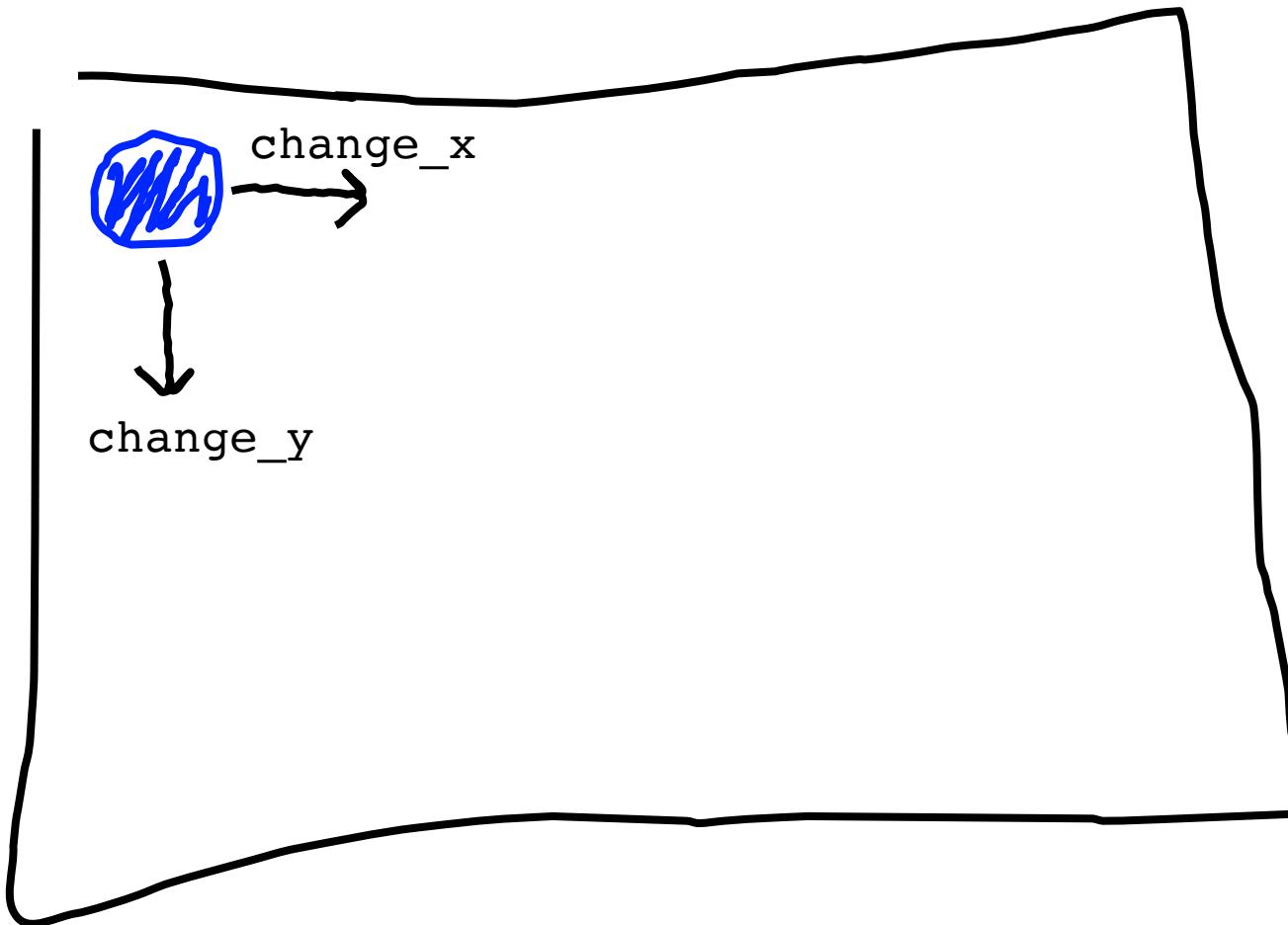


Key variable: how much the ball position
change each heartbeat?

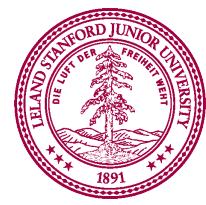


Bouncing Ball

First heartbeat

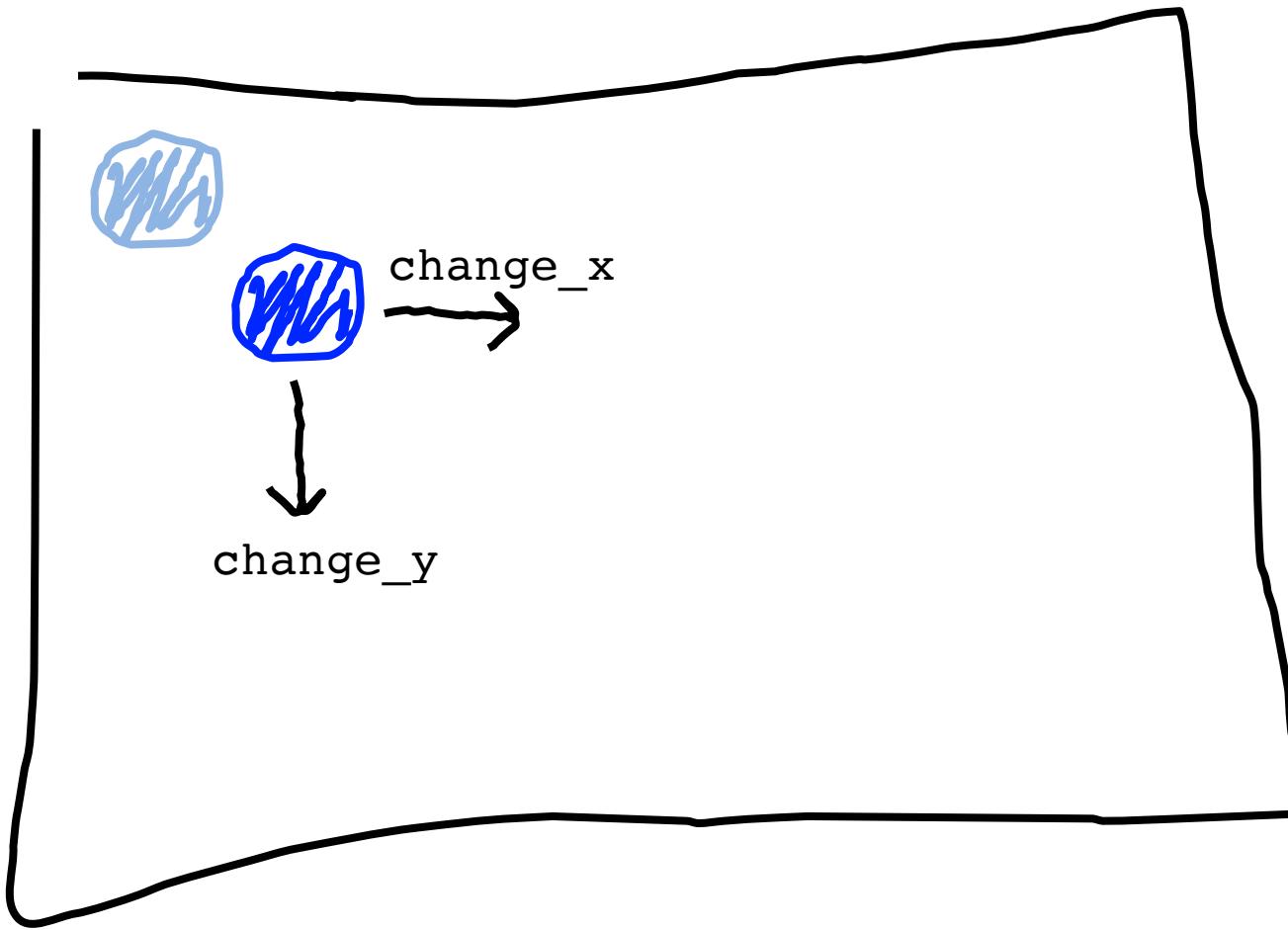


The **move** function takes in
a change in x and a change in y



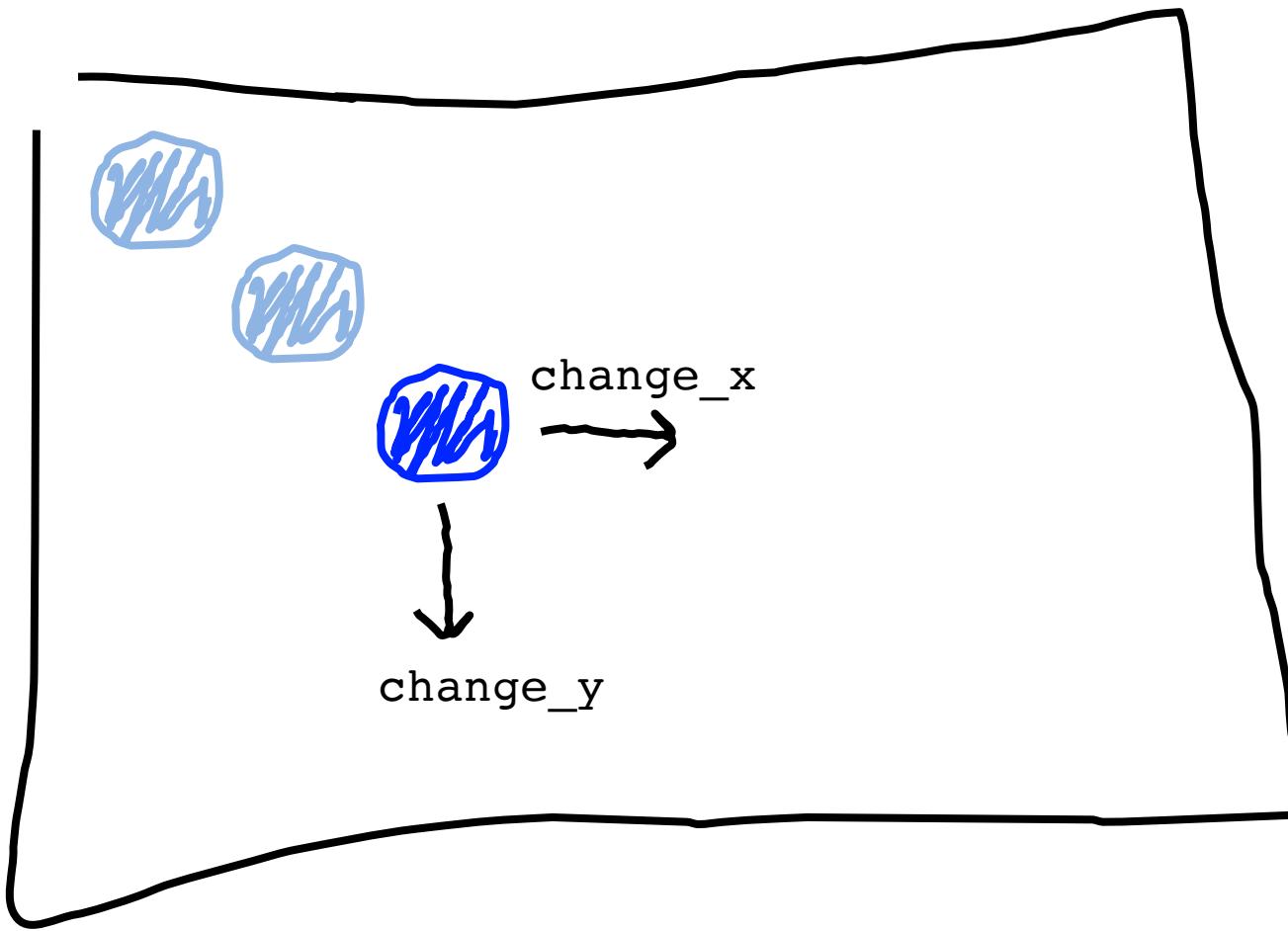
Bouncing Ball

Second heartbeat



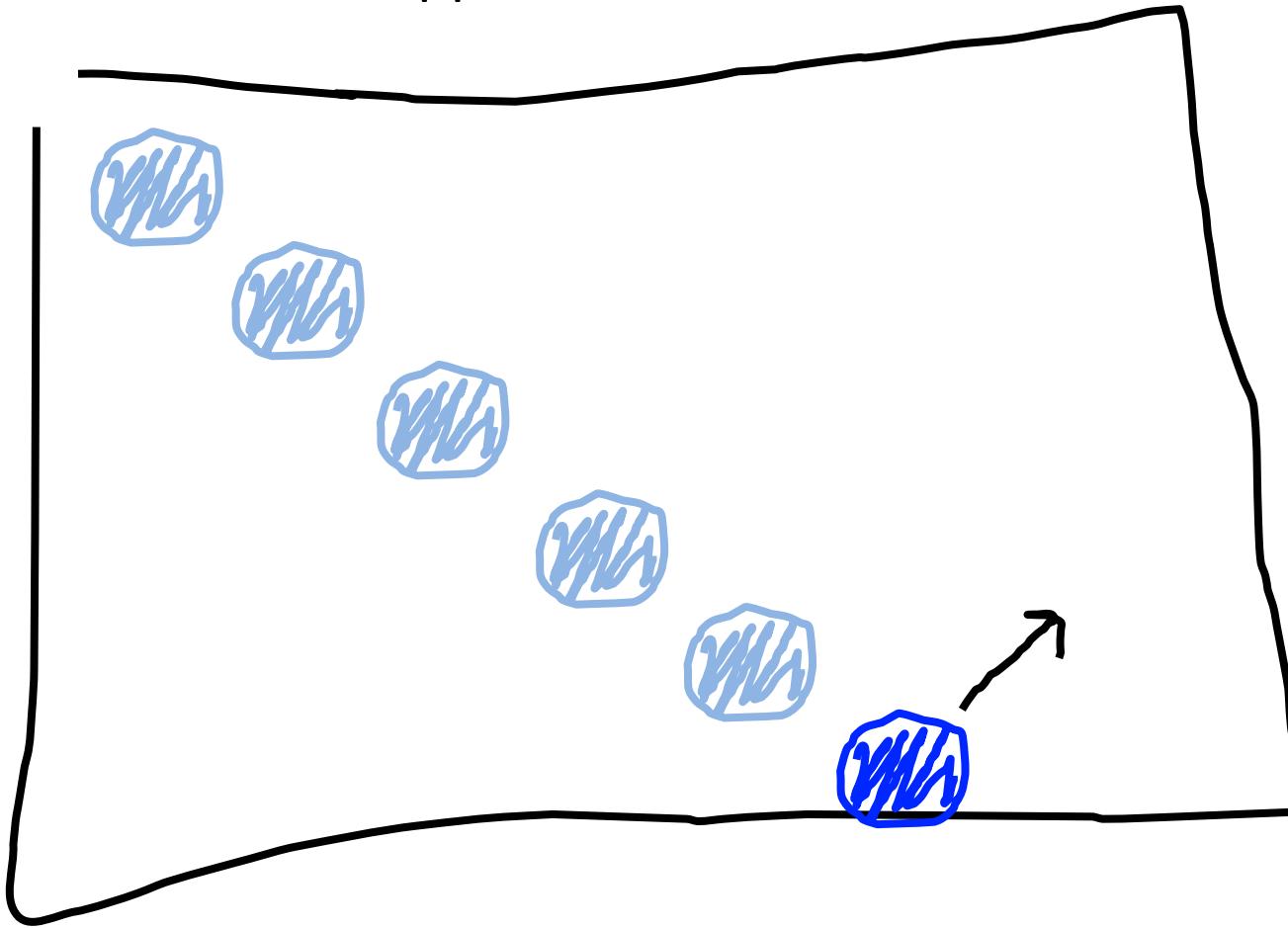
Bouncing Ball

Third heartbeat



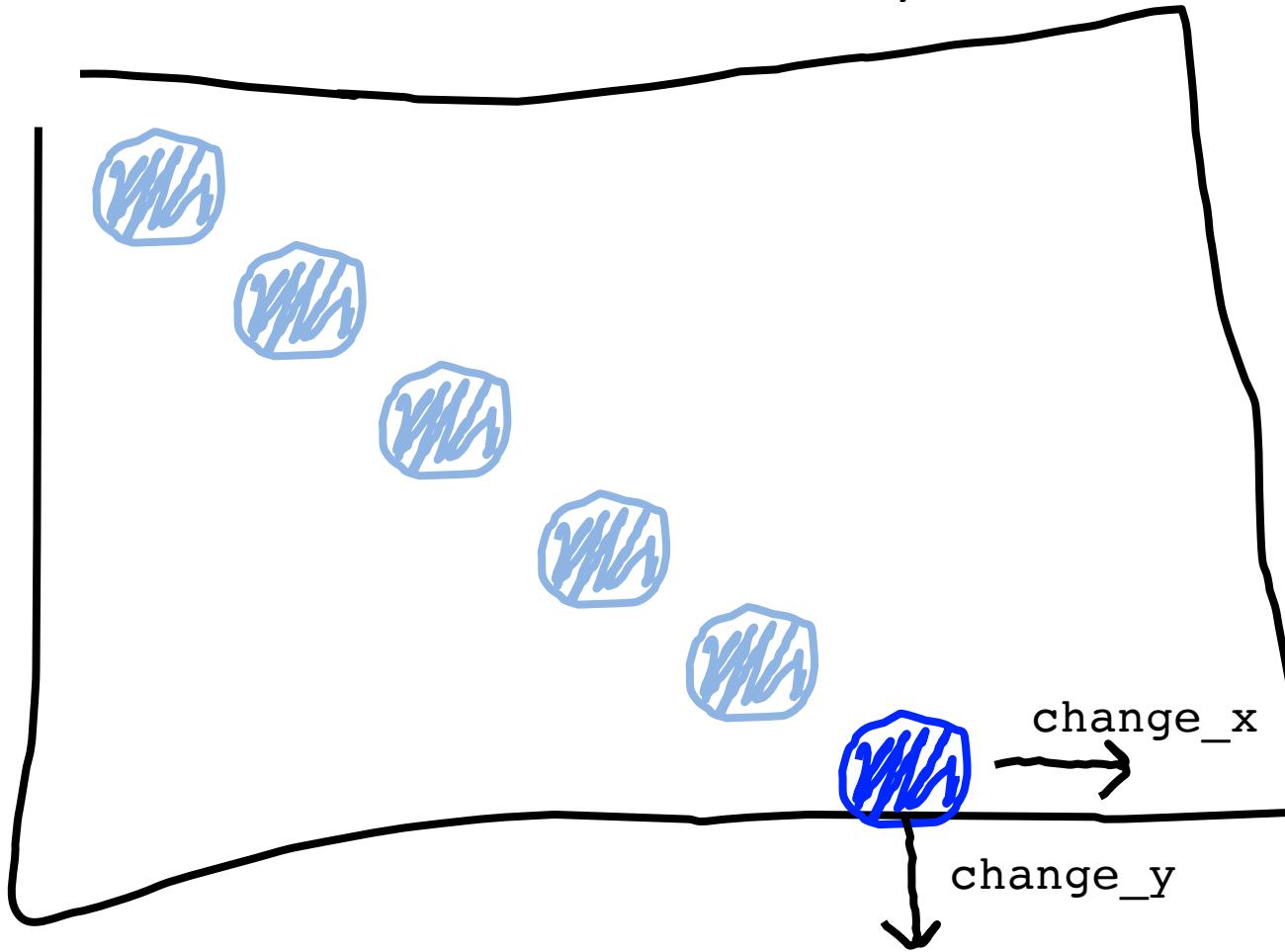
Bouncing Ball

What happens when we hit a wall?



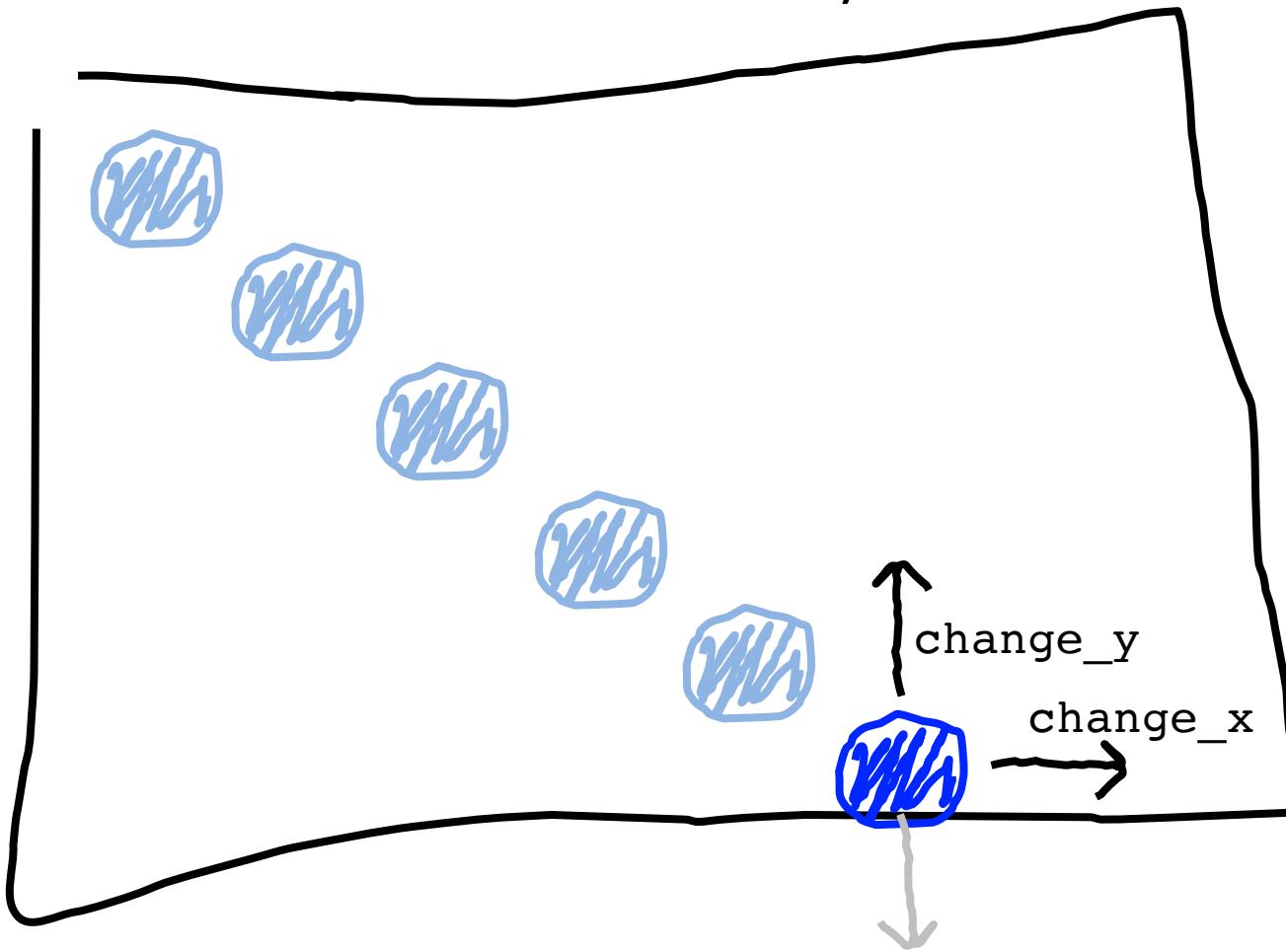
Bouncing Ball

We have this velocity



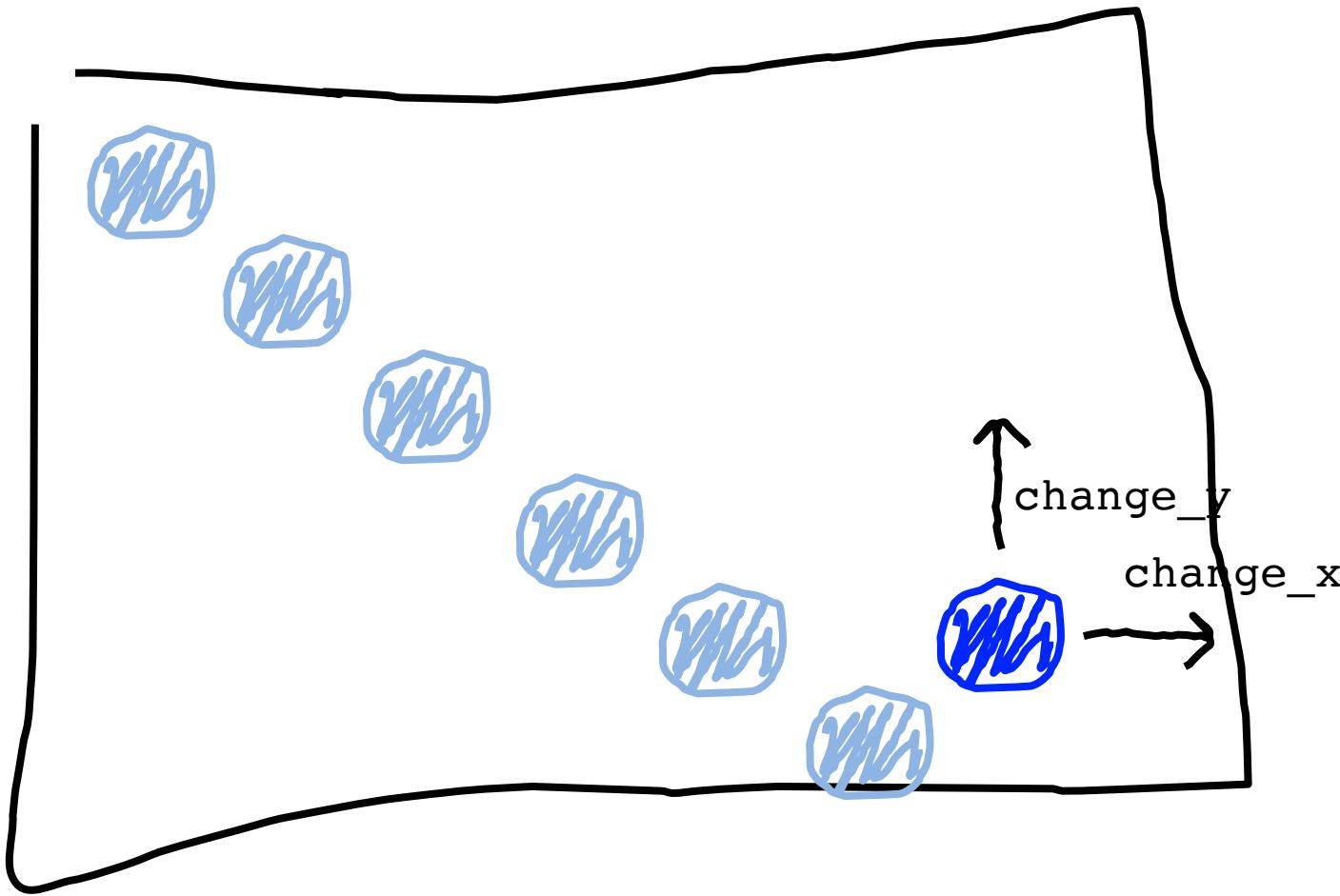
Bouncing Ball

Our new velocity



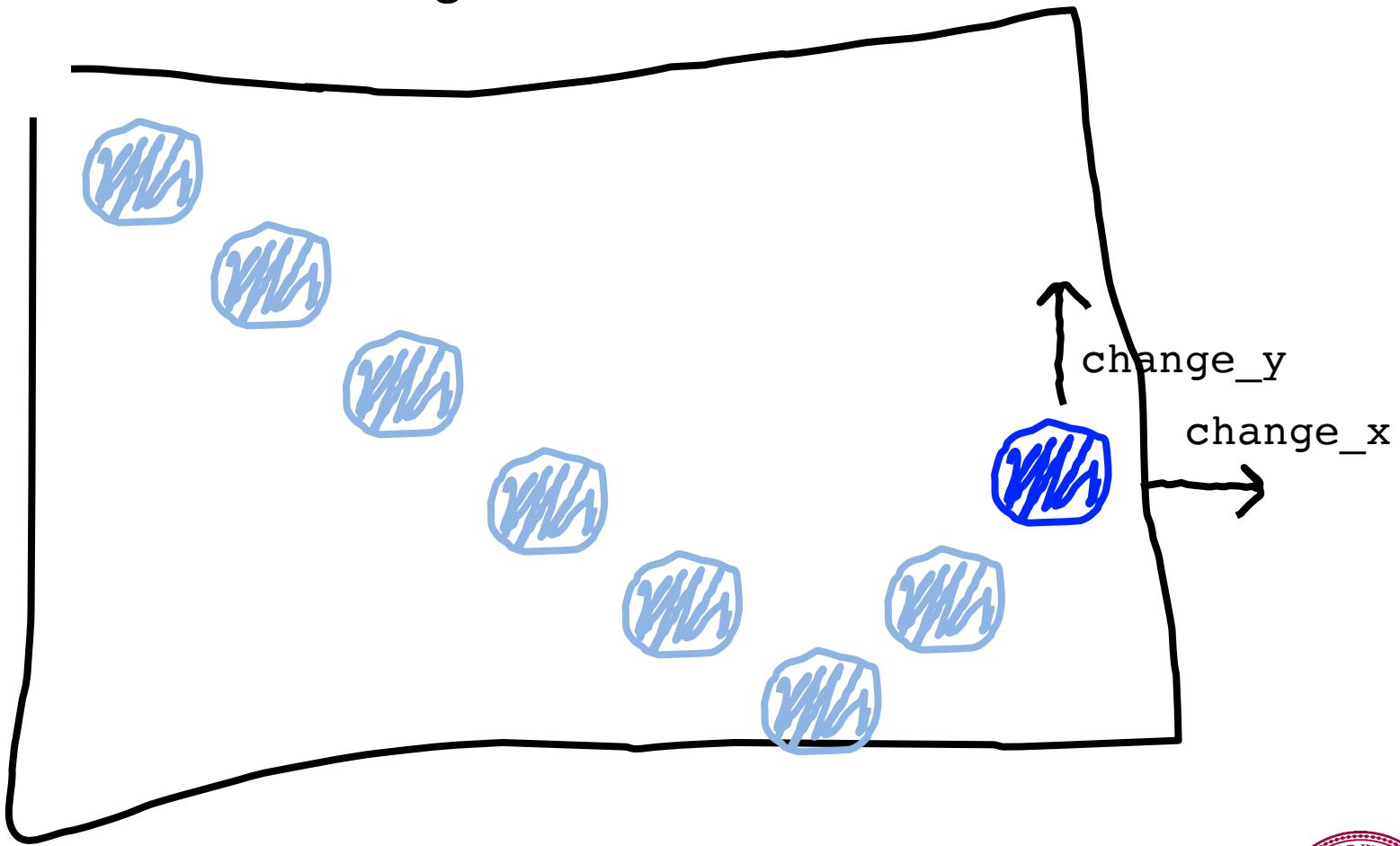
Bouncing Ball

Seventh heartbeat



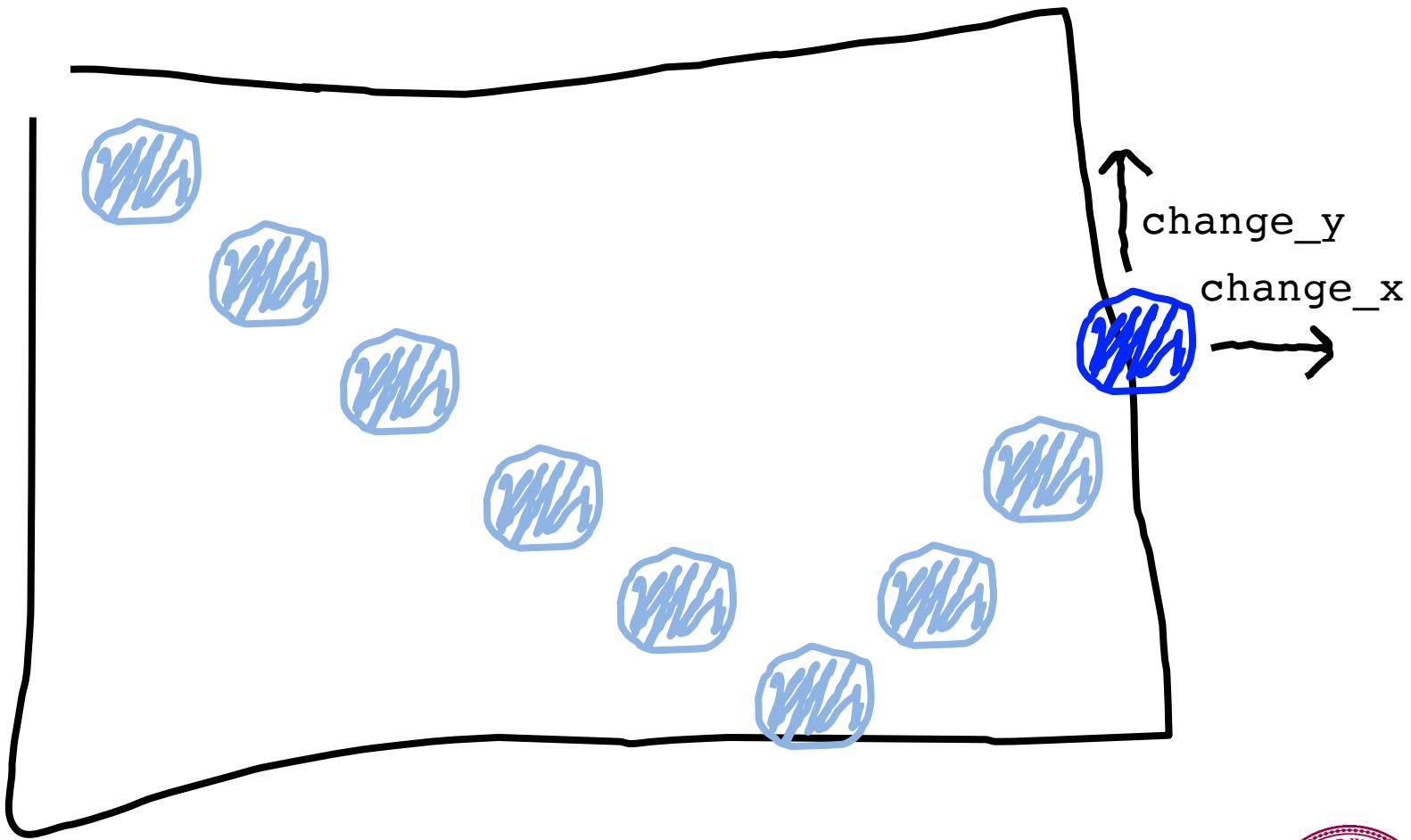
Bouncing Ball

Eighth heartbeat



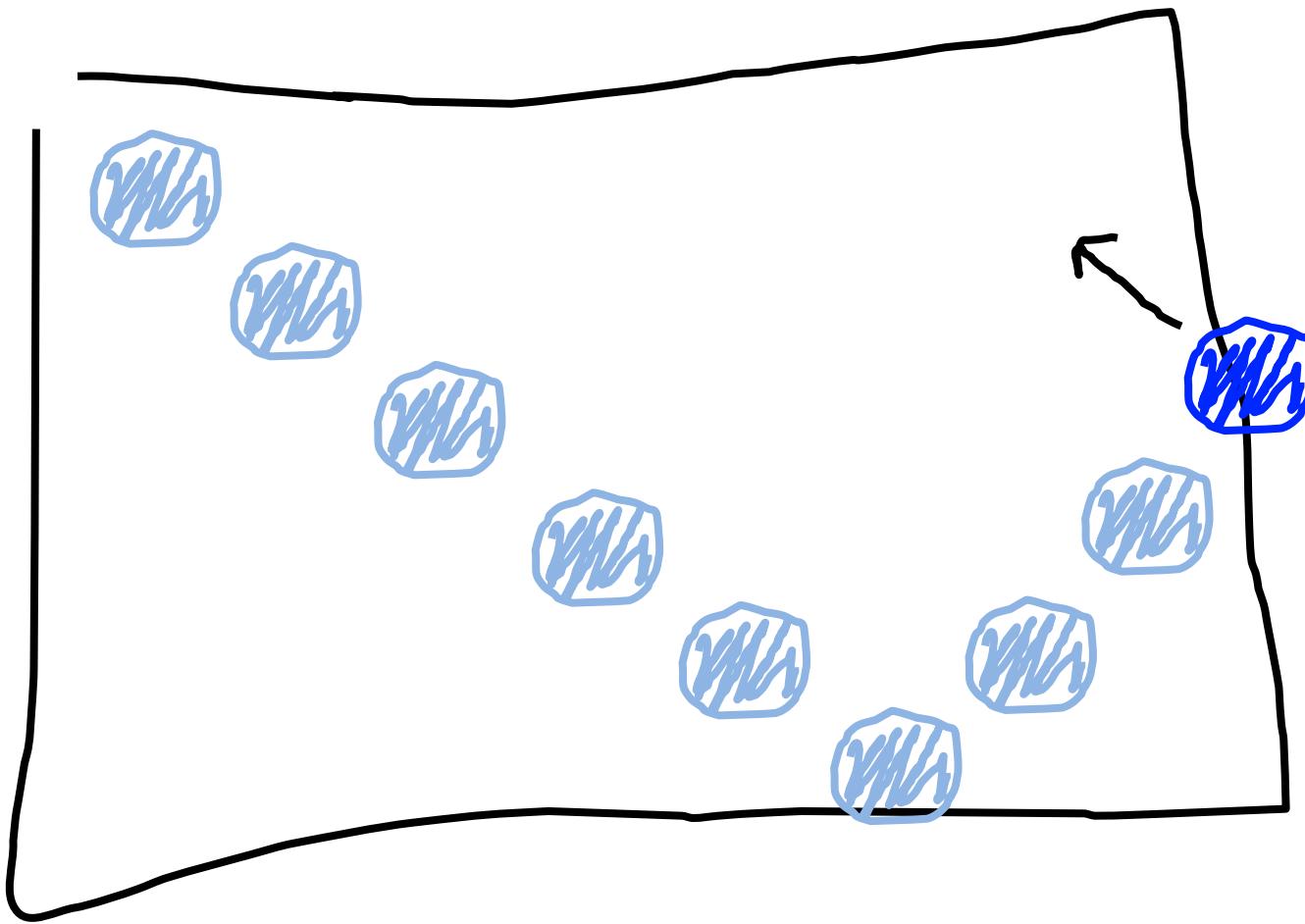
Bouncing Ball

Ninth heartbeat



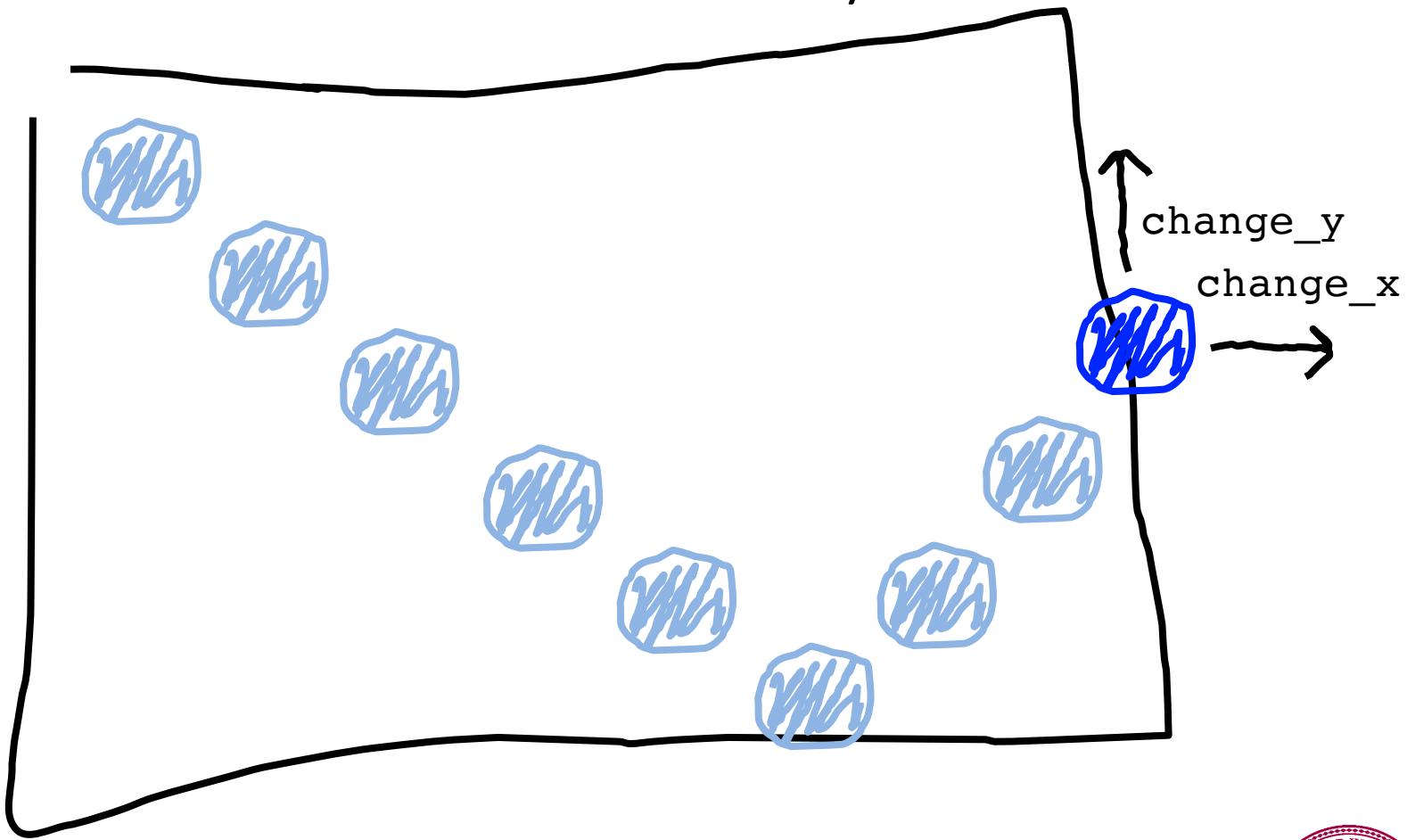
Bouncing Ball

We want this!



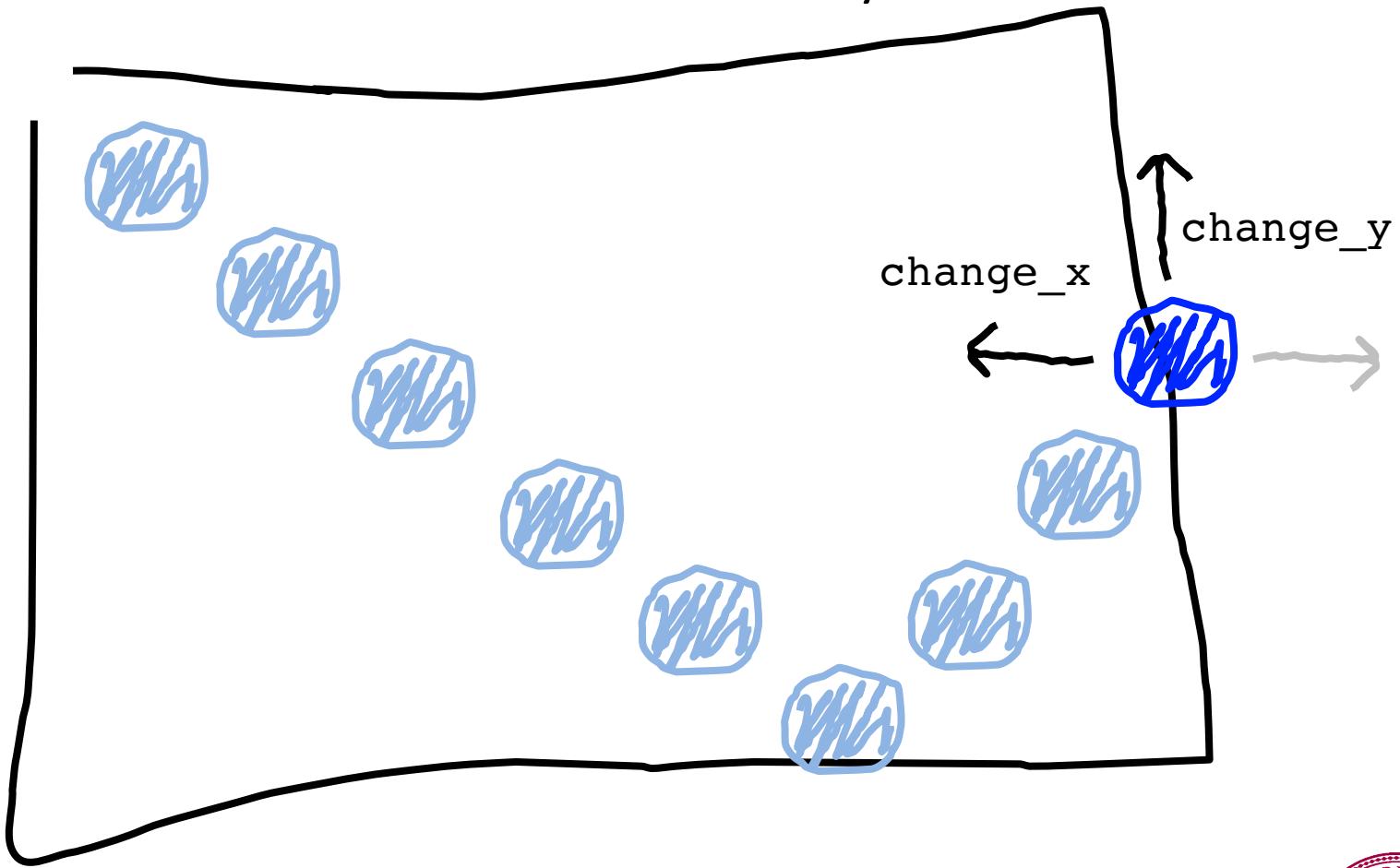
Bouncing Ball

This was our old velocity

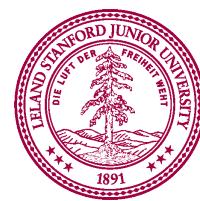


Bouncing Ball

This is our new velocity

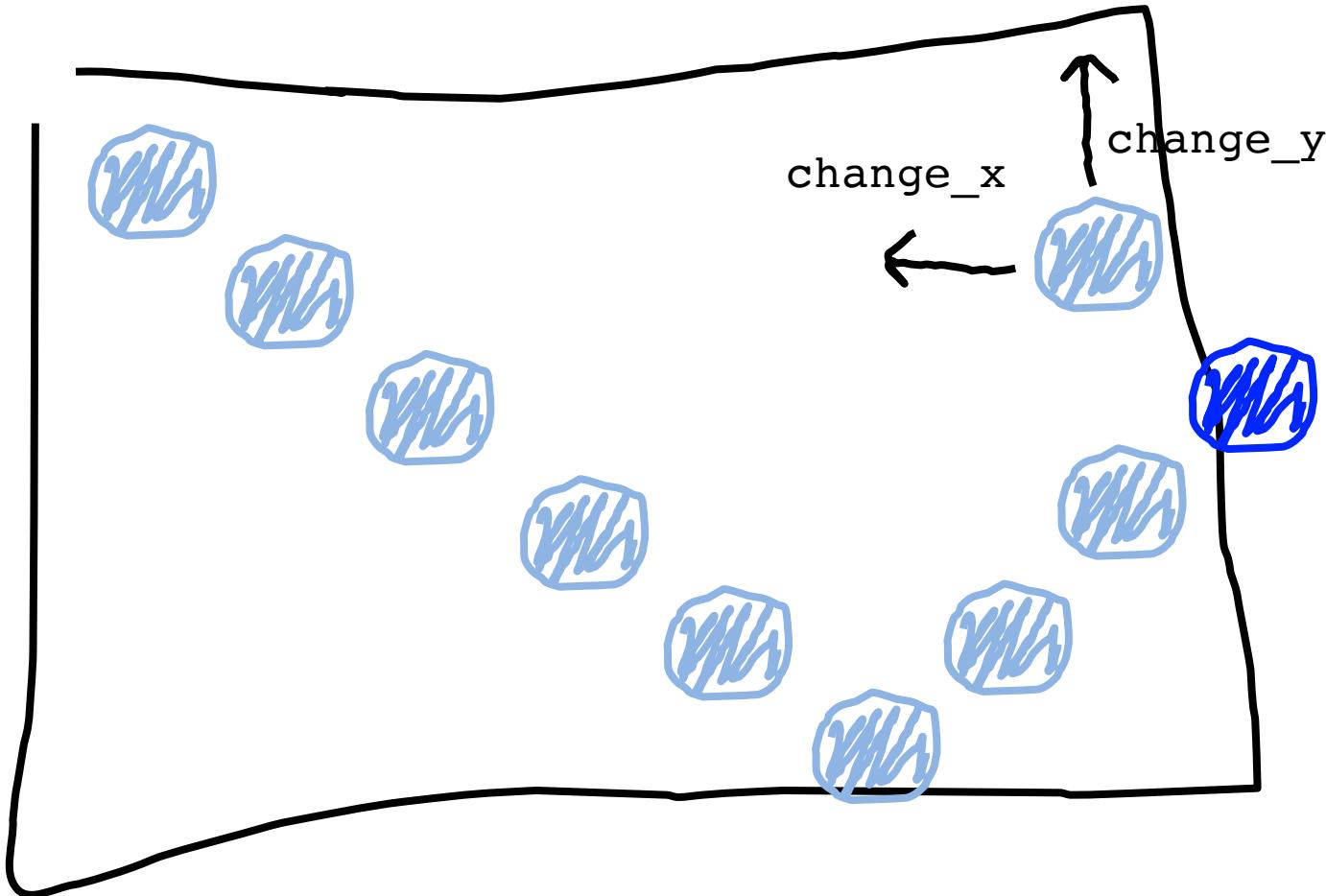


When reflecting horizontally: $\text{change}_x = -\text{change}_x$



Bouncing Ball

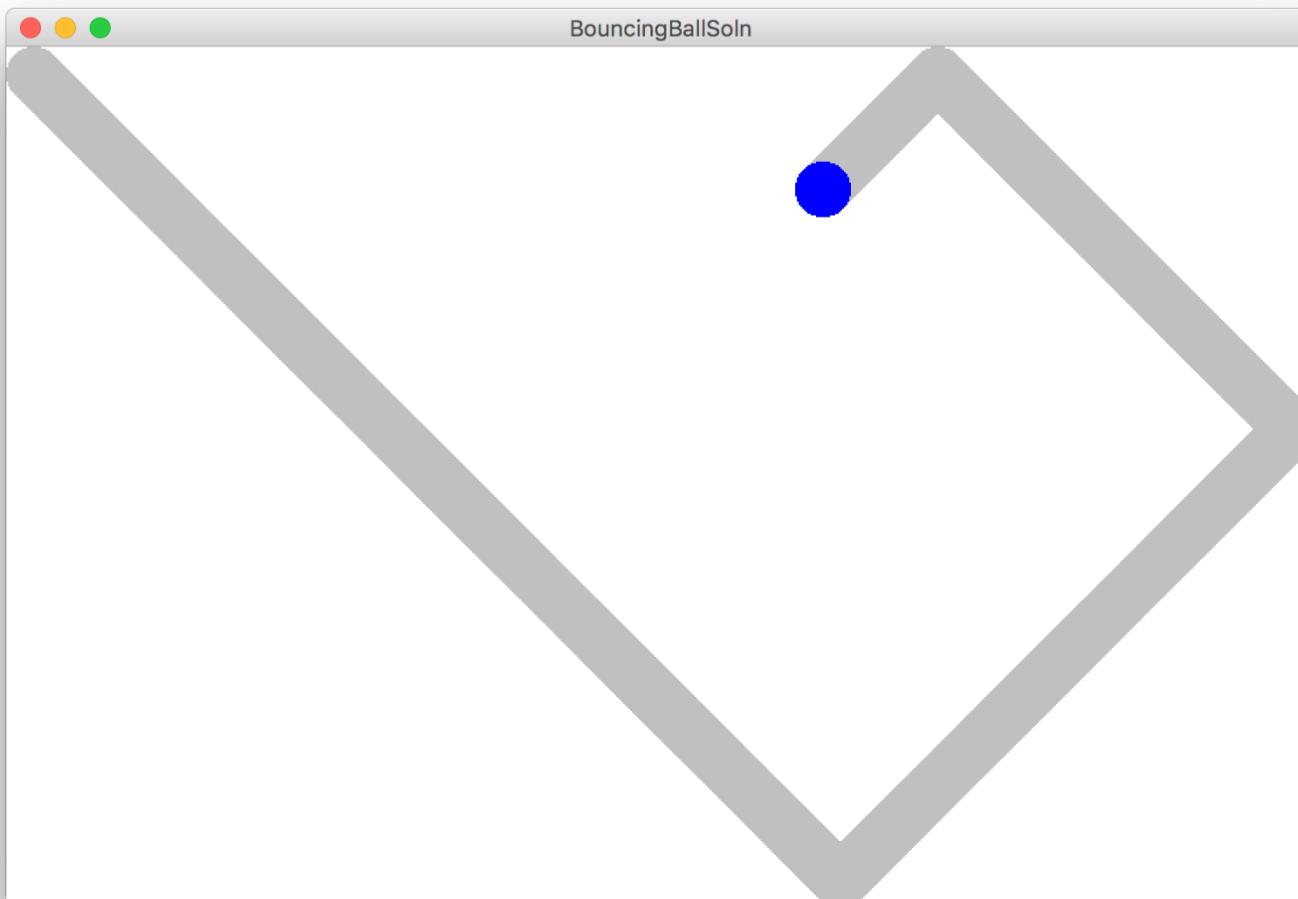
Tenth heartbeat



When reflecting horizontally: $\text{change}_x = -\text{change}_x$



Bouncing Ball

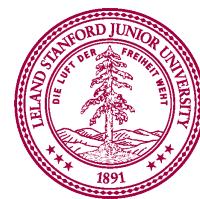


Hold up!

```
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?

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

- Title:** Apollo 11 research
- Toolbar:** Includes File, Edit, View, Insert, Format, Tools, Table, Add-ons, Help, and a note that the last edit was 3 hours ago.
- Document Outline:** On the left, it lists sections: Apollo 11 (selected), Summary, The Spacecraft, Design, Command module, Service module, The People, Neil Armstrong, Buzz Aldrin, Mission Highlights, The Launch, The Landing, and Return Trip.
- Main Content Area:**
 - Apollo 11:** The title is bolded.
 - Summary:** A paragraph about the Apollo 11 moon mission.
 - The Spacecraft:** A paragraph about the three spacecraft: Command Module Columbia, Service Module, and Lunar Module Eagle.
 - Design:** A paragraph about the key NASA spacecraft involved in the mission: Saturn V rocket, Apollo CSM-107, and Apollo LM-5.
 - Command module:** A paragraph about the Command/Service Module (CSM) and its role in the Apollo program.
 - Service module:** A paragraph about the Service Module's function and the engine used for the Moon landing.
- Right Side:** Includes a 'Comments' button and a 'Share' button.

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



```
def main():
```

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

```
    make_ball(canvas)
```

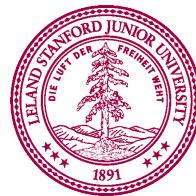
```
def make_ball(canvas):
```

```
    canvas.create_rectangle( ... , fill='blue')
```

stack

heap

```
main
```



```
def main():
```

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

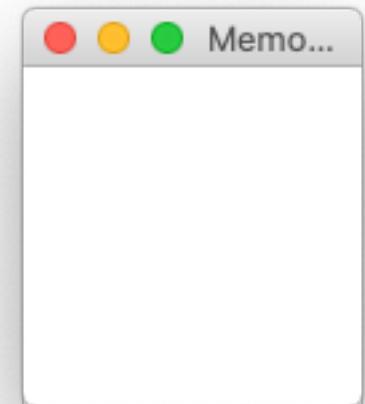
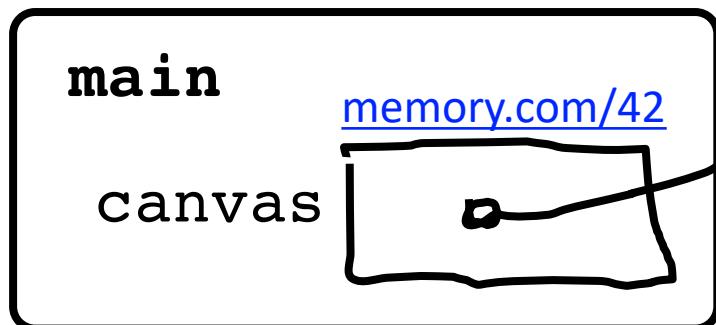
```
    make_ball(canvas)
```

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

```
    canvas.create_rectangle( ... , fill='blue')
```

stack

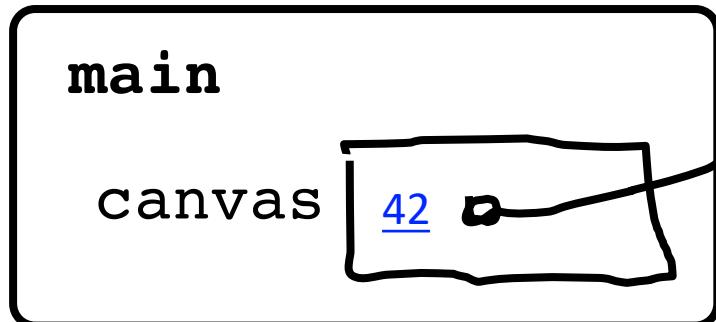
heap



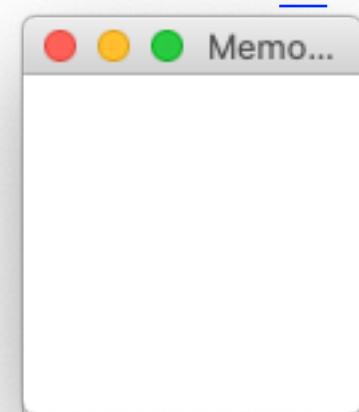
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

stack



heap



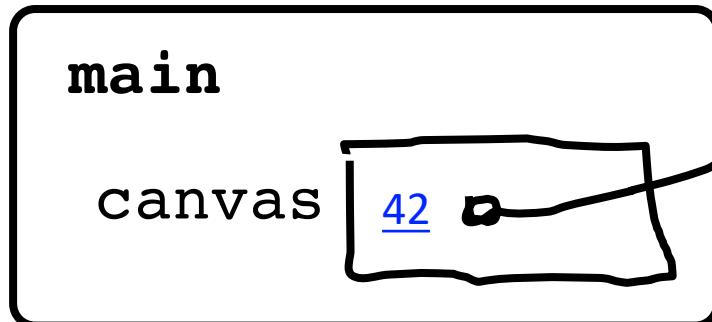
42



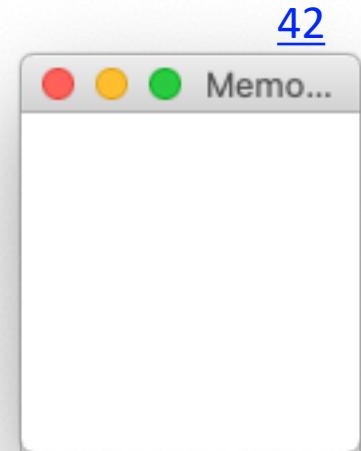
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

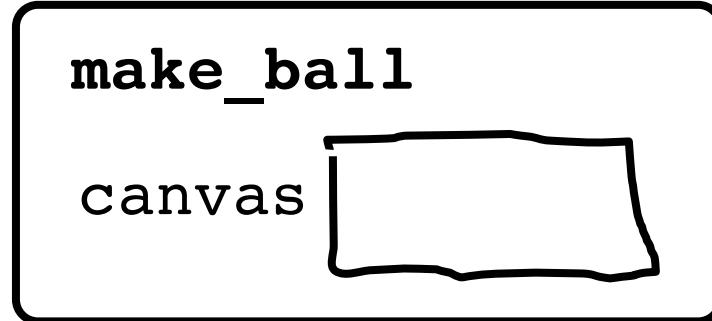
stack



heap



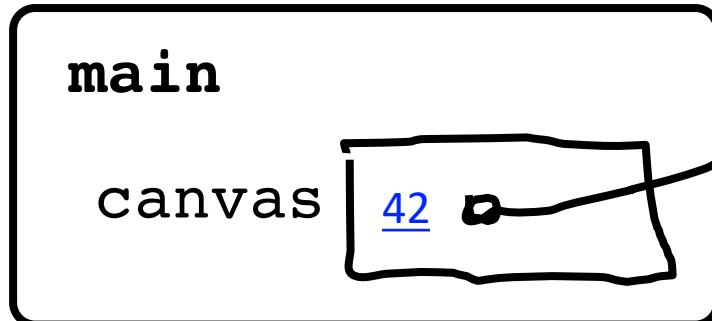
make_ball



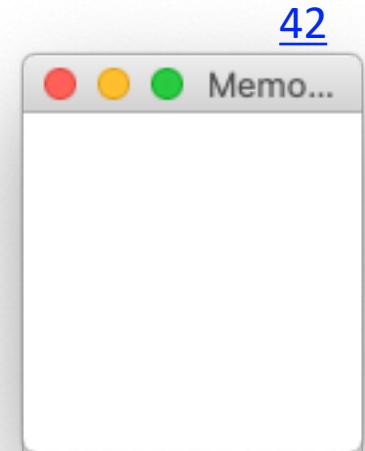
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

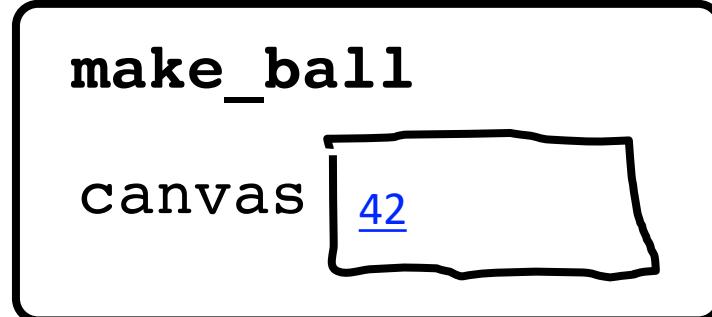
stack



heap



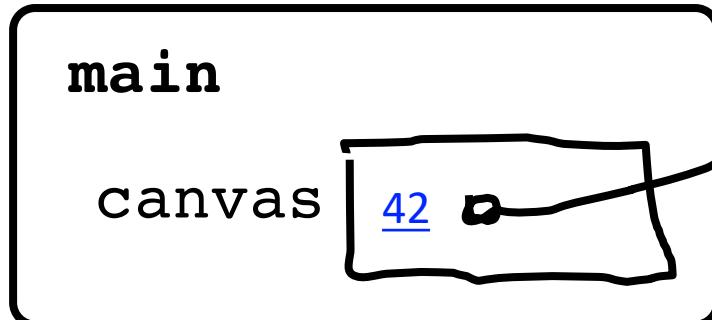
make_ball



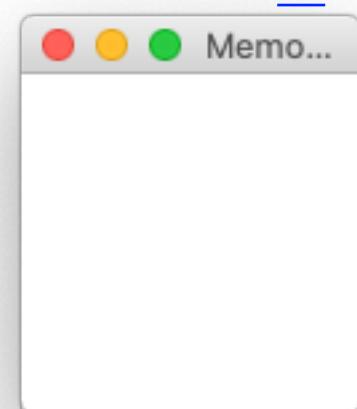
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

stack

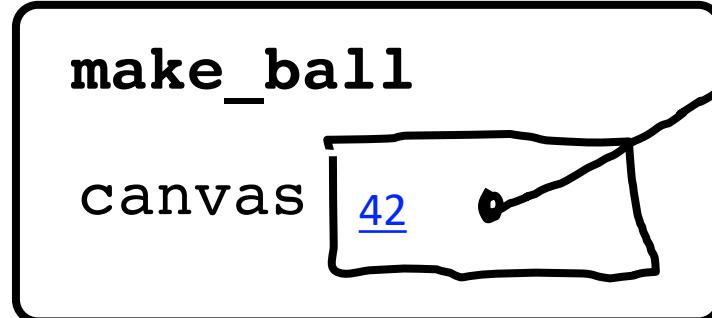


heap



[42](#)

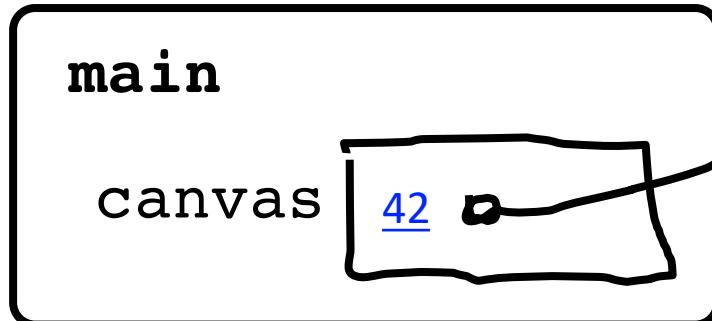
make_ball



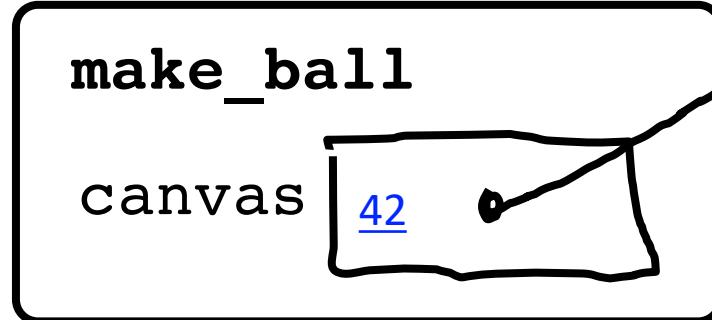
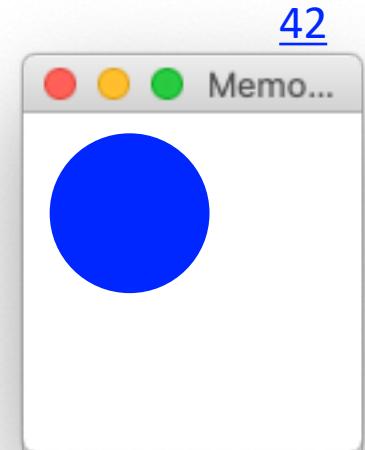
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

stack



heap

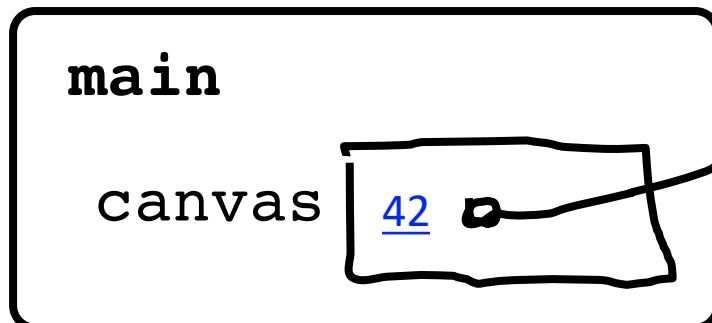


```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)

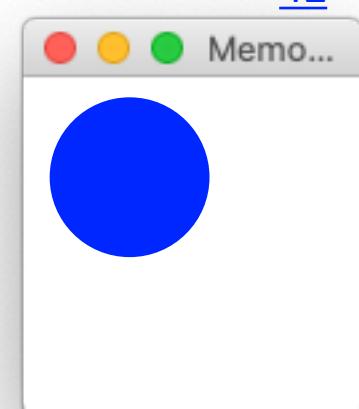
def make_ball(canvas):
    canvas.create_oval( ... , fill='blue')
```



stack



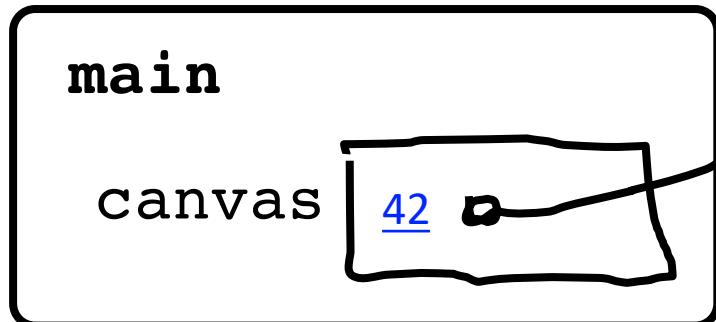
heap



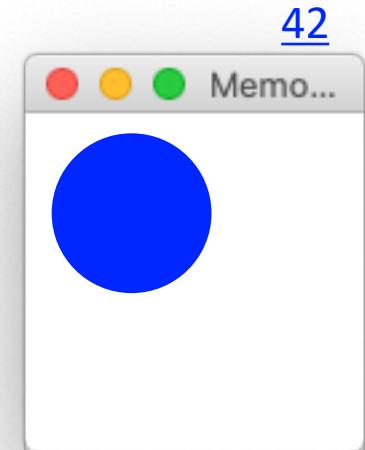
```
def main():
    canvas = make_canvas(...)
    make_ball(canvas)
```

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

stack



heap





Some variables are stored
with references

(which are like memory
URLs)



When passed as parameters

Variables that act like
they are **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!



