15-112 Lecture 2

Animations

Instructor: Pat Virtue

Tuesday Logistics

As you walk in

Quiz will start at the beginning of lecture

- Have pencil/pen ready
- Silence phones

Quiz

Before we start

- Don’t open until we start
- Make sure your name and Andrew ID are on the front
- Read instruction page
- No questions (unless clarification on English)

Additional info

- 25 min

Announcements

Same as usual ☺

HW 4 Tips

VS Code set up

- Read instructions carefully

Autograder with animations

- Read piazza post
- Video different than current CS Academy autograder

Start early

- It’s a lot of code to write

Thursday Logistics

Announcements

Quiz 3 debrief

- 5 point bump applied in Gradescope

Unit 3 Sections

- 3.7-3.9 Helpful (and fun!) but won't block your from proceeding to exercises
  - You should still be able to do basic bounded motion though, 3.8

Basic Shapes

Rectangles, Ovals, Lines, and Labels

Reference slide

```
drawRect(x0, y0, width, height)

drawOval(centerX, centerY, width, height)

drawLine(x0, y0, x1, y1)

drawLabel(text, centerX, centerY)
```

Rectangles, Ovals, Lines, and Labels

Reference slide

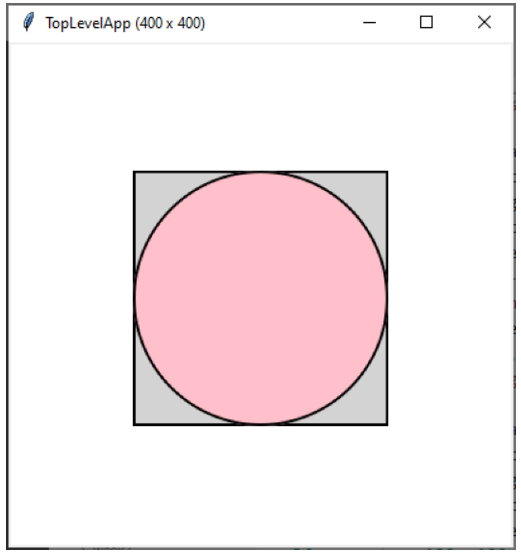
```
drawRect(x0, y0, width, height, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0)

drawOval(centerX, centerY, width, height, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0)

drawLine(x0, y0, x1, y1, fill='black', lineWidth=2, opacity=100,
    dashes=False, arrowStart=False, arrowEnd=False)

drawLabel(text, centerX, centerY, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0,
    size=12, font='arial', bold=False, italic=False, align='center')
```

Poll 1

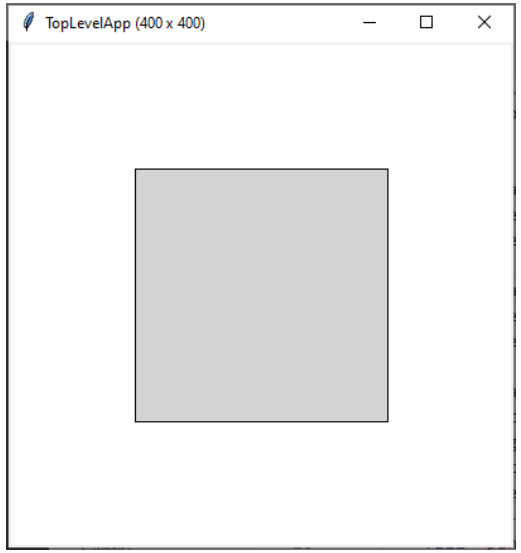
Which does this draw?

```
def drawBubble(app):
    drawOval(200, 200, 200, 200, fill='pink', border='black')
    drawRect(100, 100, 200, 200, fill='lightgray', border='black')
```

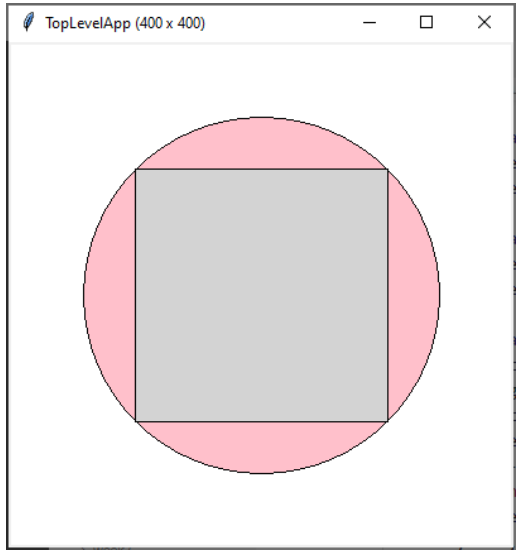
A.



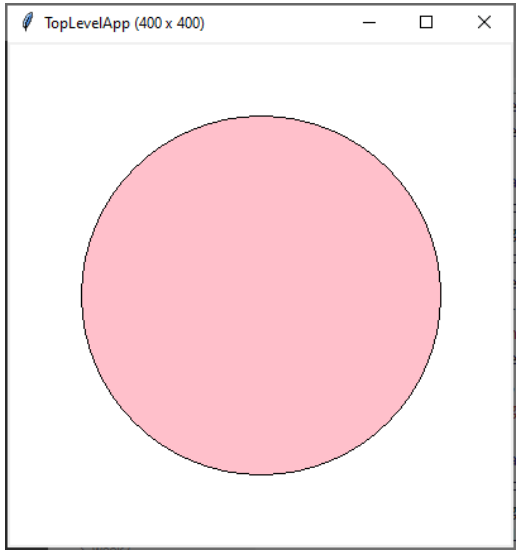
B.



C.



D.



E. I have no idea

Tip: Create Sandbox Cheatsheets for Yourself!

3.3.4 Labels

Label Alignment

This next example shows all the values you can use for alignment:

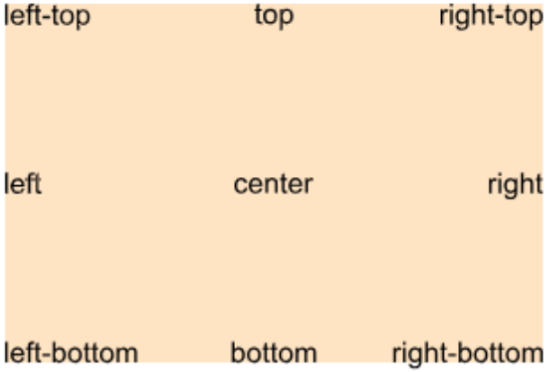
```
from cmu_graphics import *

def redrawAll(app):
    drawRect(50, 100, 3
    drawLabel('Left-Top
    drawLabel('Left', 5
    drawLabel('Left-Bot

    drawLabel('Top', 20
    drawLabel('Center',
    drawLabel('Bottom',

    drawLabel('Right-To
    drawLabel('Right',
    drawLabel('Right-Bo

def main():
    runApp()

main()
```

align_demo.py

Run

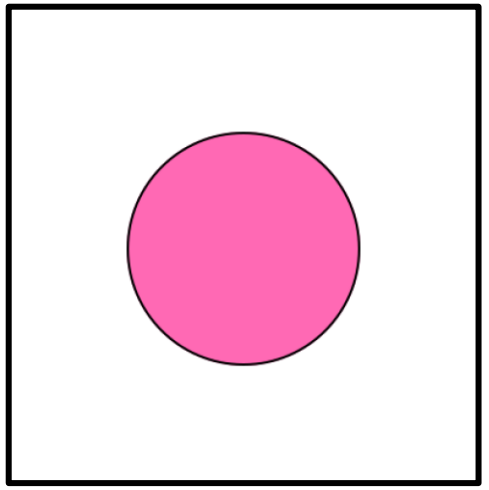
Saved

```
from cmu_graphics import *

def redrawAll(app):
    drawRect(50, 100, 300, 200, fill='bisque')
    drawLabel('left-top', 50, 100, align='left
    drawLabel('left', 50, 200, align='left', s
    drawLabel('left-bottom', 50, 300, align='l

    drawLabel('top', 200, 100, align='top', si
    drawLabel('center', 200, 200, align='cente
    drawLabel('bottom', 200, 300, align='botto

    drawLabel('right-top', 350, 100, align='ri
    drawLabel('right', 350, 200, align='right'
    drawLabel('right-bottom', 350, 300, align=

def main():
    runApp()

main()
```



Poll 2

Which of these is best?



Left

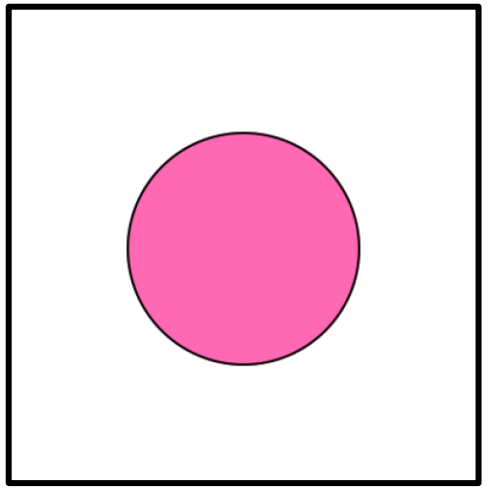
```
def drawBubbleA(app):
  drawOval(200, 200, 200, 200,
    fill='hotpink', border='black')
```

Right

```
def drawBubbleB(app):
  cx = 200
  cy = 200
  width = 200
  height = 200
  drawOval(cx, cy, width, height,
    fill='hotpink', border='black')
```

Poll 3

Which of these is best?



Left

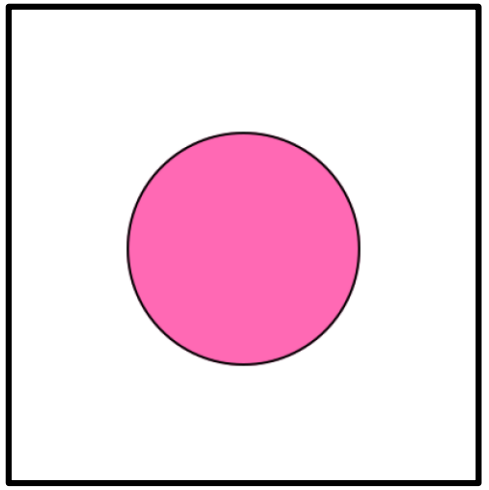
```
def drawBubbleC(app):
  cx = app.width/2
  cy = app.height/2
  d = app.width/2
  drawOval(cx, cy, d, d,
    fill='hotpink', border='black')
```

Right

```
def drawBubbleB(app):
  cx = 200
  cy = 200
  width = 200
  height = 200
  drawOval(cx, cy, width, height,
    fill='hotpink', border='black')
```

Poll 4

Which of these is best?



Left

```
def drawBubbleC(app):
  cx = app.width/2
  cy = app.height/2
  d = app.width/2
  drawOval(cx, cy, d, d,
    fill='hotpink', border='black')
```

Right

```
def drawBubbleD(app):
  cx = app.width/2
  cy = app.height/2
  d = min(app.width/2, app.height/2)
  drawOval(cx, cy, d, d,
    fill='hotpink', border='black')
```

Exercise: Bigger Smaller

Guided exercise in notes

Poll 5

Exercise: Bigger Smaller

(This was not explicitly part of the pre-reading, I just want to know how much I'm repeating things from the notes)

What happens when I click on the white space outside the red square? Select all that apply

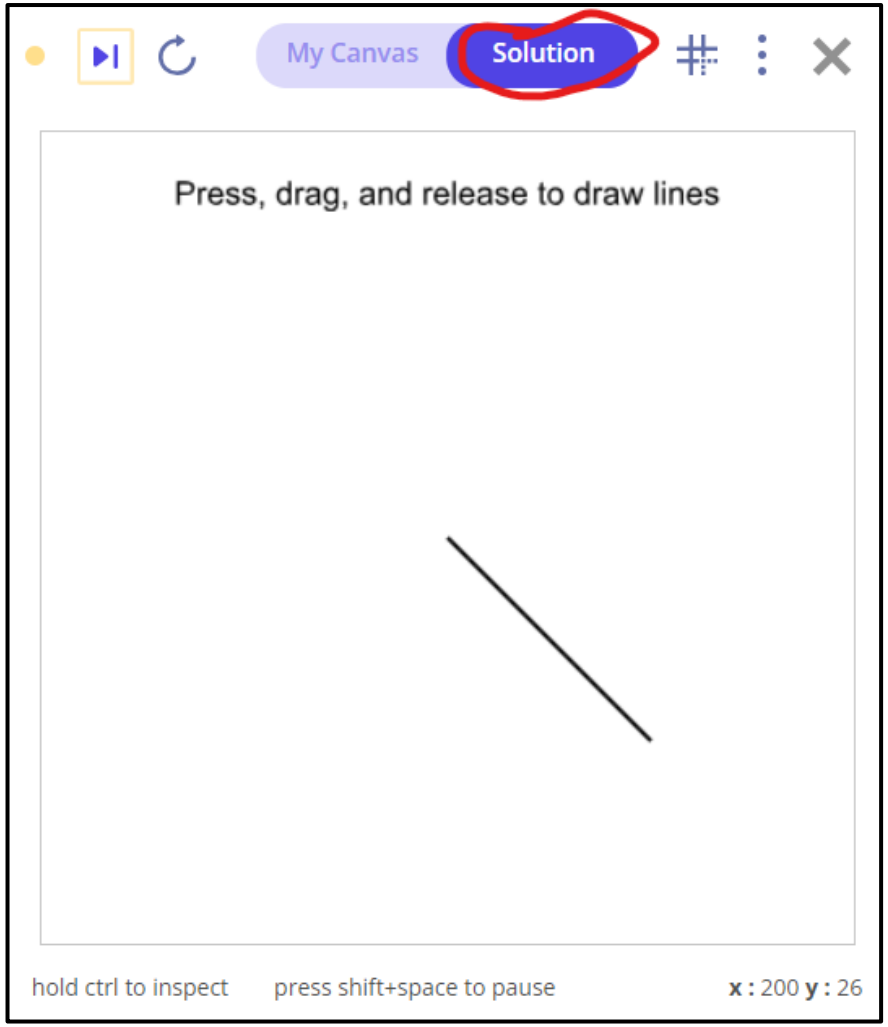
A. Nothing (Hint: it isn't this one. Do NOT select this one)
B. Square gets bigger
C. Square gets smaller
D. Square turns green
E. Square turns pink

Rectangles, Ovals, Lines, and Labels

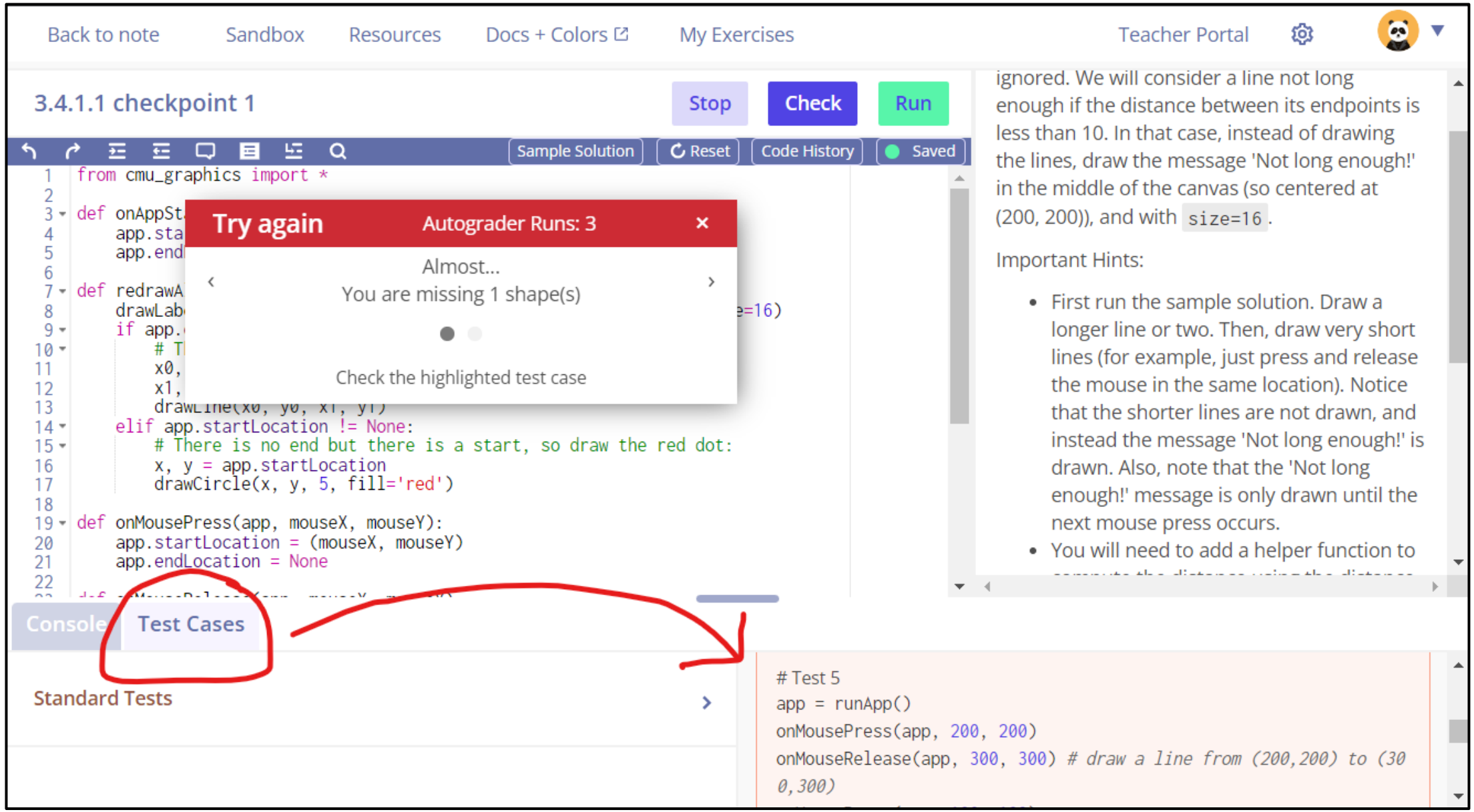
```
drawRect(x0, y0, width, height, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0)

drawRect(x?, y?, width, height, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0,
    align='???’)

drawRect(centerX, centerY, width, height, fill='black', opacity=100,
    border=None, borderWidth=2, rotateAngle=0,
    align='center')
```

CS Academy Autograder

CS Academy Autograder with Animations



CS Academy Autograder with Animations



Events

Another good reference for the sandbox

App Skeleton

Tip: Super convenient for sandbox

Note: Next quiz we'll provide function names

Important: Don't call event functions or redrawAll!

appSkeleton.py

```
1  from cmu_graphics import *
2
3  ### Controller
4
5  def onAppStart(app):
6      app.paused = True
7      # app.stepsPerSecond = 30
8
9  def onStep(app):
10     if not app.paused:
11         takeStep(app)
12
13 def takeStep(app):
14     pass
15
16 def onKeyPress(app, key):
17     if key == 's' and not app.paused:
18         takeStep(app)
19
20     elif key == 'p':
21         app.paused = not app.paused
22
```

```
23 def onKeyRelease(app, key):
24     pass
25
26 def onKeyHold(app, key):
27     pass
28
29 def onMousePress(app, mouseX, mouseY):
30     pass
31
32 def onMouseRelease(app, mouseX, mouseY):
33     pass
34
35 def onMouseDrag(app, mou
36     pass
37
38 def onMouseMove(app, mou
39     pass
40
41 def onResize(app):
42     pass
43
```

```
44 ### View
45
46 def redrawAll(app):
47     pass
48
49 ### Main
50
51 def main():
52     runApp()
53
54 main()
55
```

Debugging:

Even more debugging options with graphics

- Good old print debugging still works

```
def onAppStart(app):
    print('onAppStart')
    app.text = ''

def onKeyPress(app, key):
    print(f'onKeyPress: {key} ')
    app.text += key


def redrawAll(app):
    print('REDRAW ALL')
    drawLabel(app.text, app.width/2,
        app.height/2, size=50)
```

- Can also create a temporary debug label: drawLabel(app.debugText, …)

Poll 6

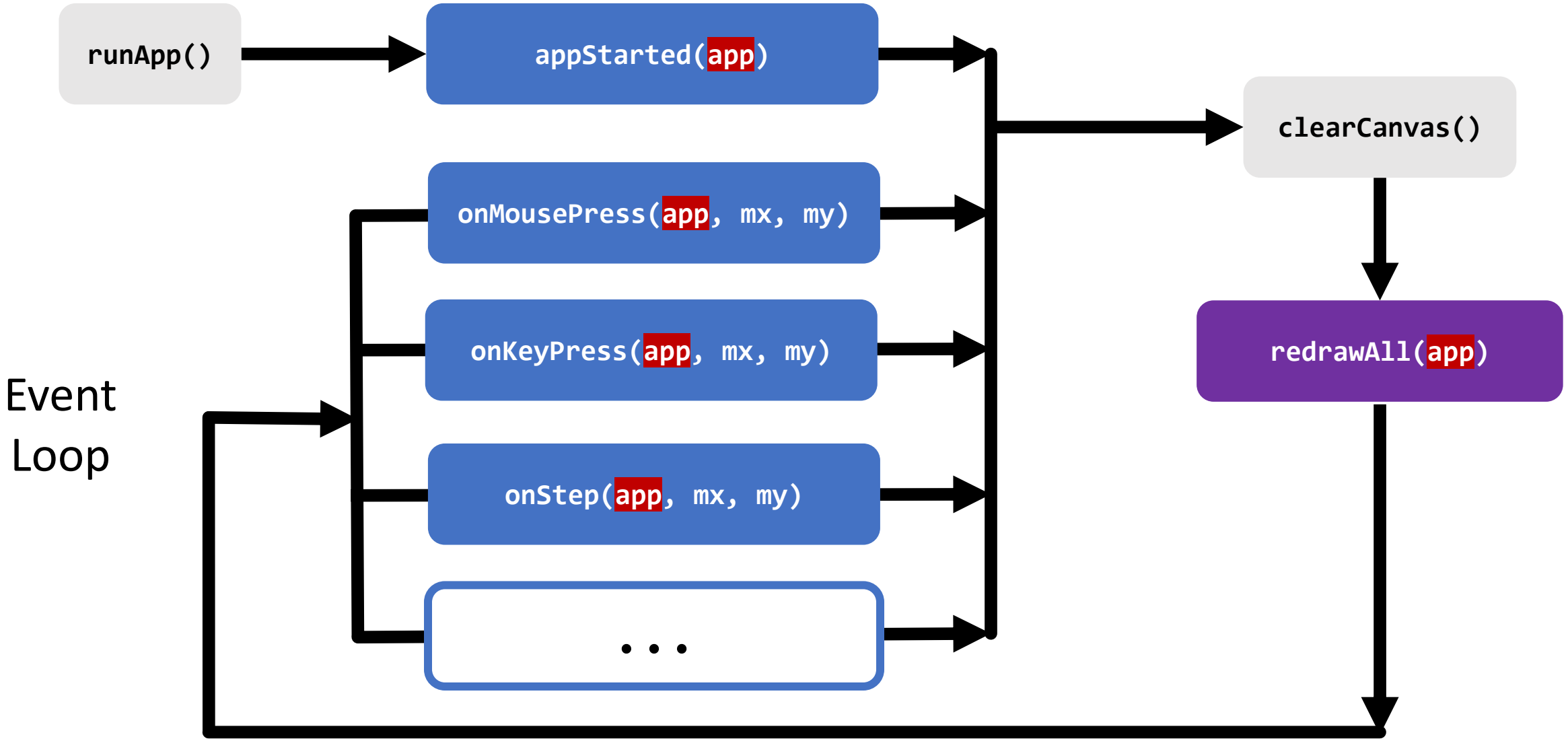
How many times will REDRAW ALL print after I start the app and press the letters A and then Q?

A. 0
B. 1
C. 2
D. 3
E. 4
F. 5
G. A lot!
H. I have no idea

```
def onAppStart(app):
    print('onAppStart')
    app.text = ''

def onKeyPress(app, key):
    print(f'onKeyPress: {key}')
    app.text += key

def redrawAll(app):
    print('REDRAW ALL')
    drawLabel(app.text, app.width/2,
        app.height/2, size=50)
```

Model-View-Controller

MVC (Model-View-Controller)

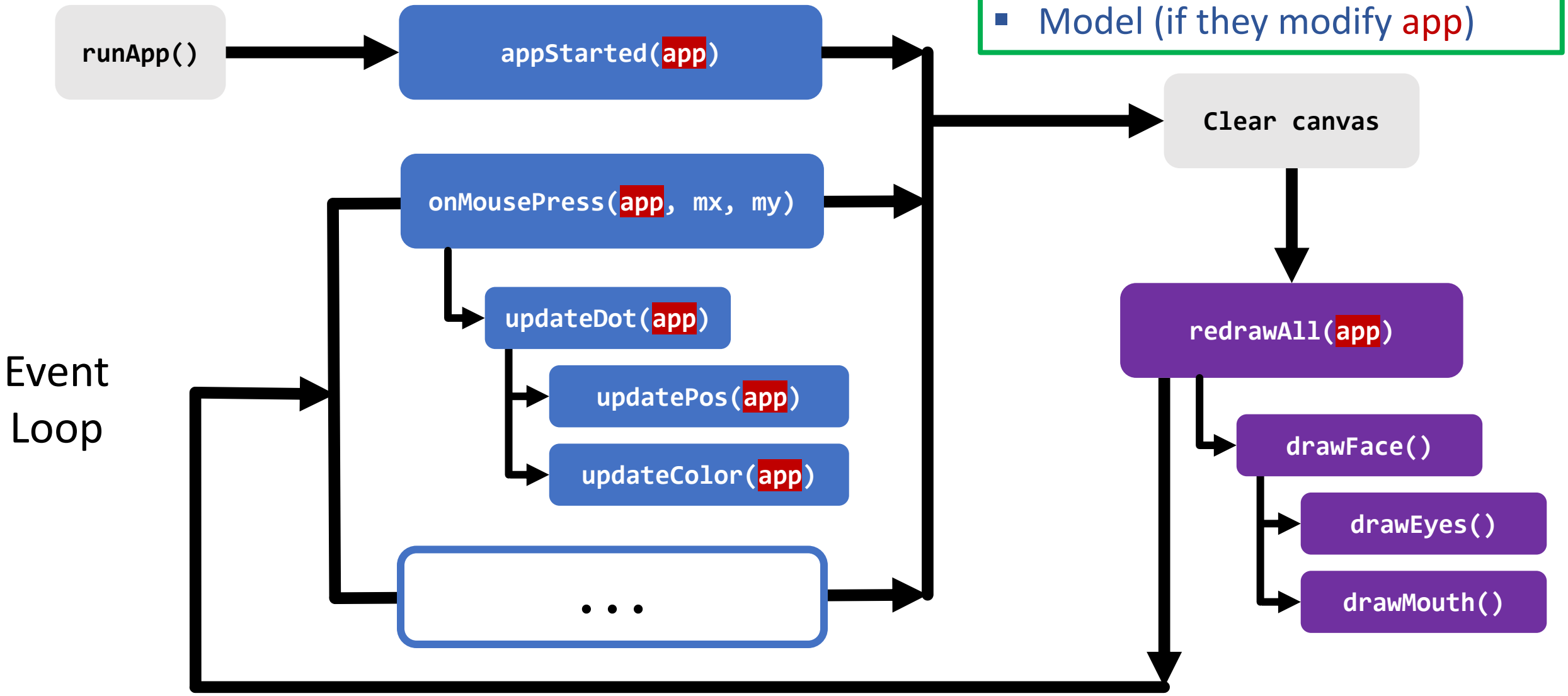


MVC (Model-View-Controller)

Important!

Helper functions are part of:

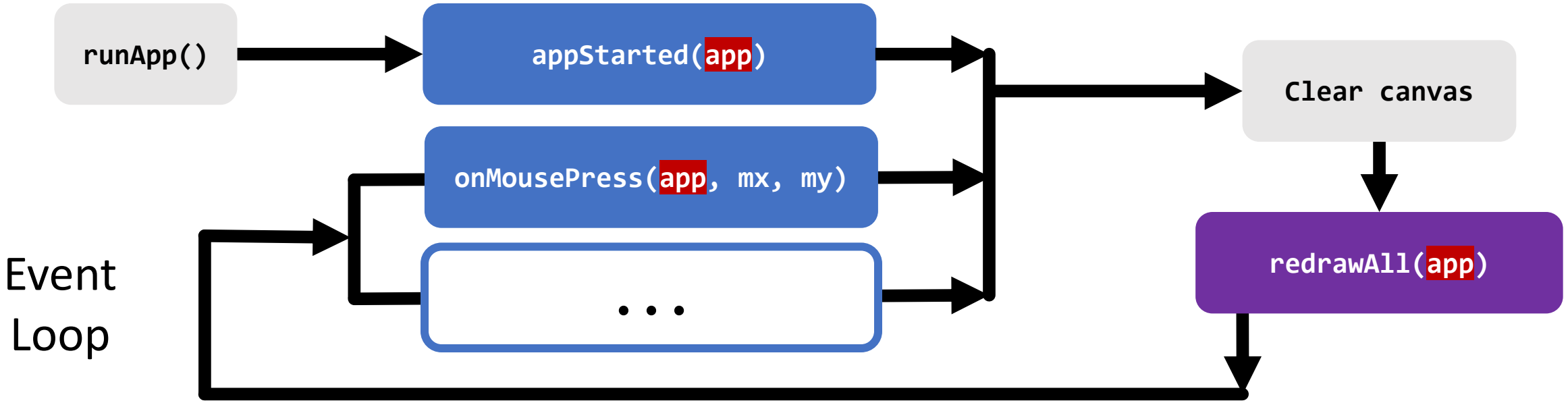
- View (if they draw anything)
- Model (if they modify app)



MVC (Model-View-Controller)

Rules

- View cannot change the Model (app)
- Controller cannot change the View (redrawAll or draw***)



Poll 7

Which of the following are true?

Select all that apply

A. onMousePress and onKeyPress can run at the exact same time

B. redrawAll and onKeyPress can run at the exact same time

C. Python crashes if we return a value in onKeyPress

D. None of the above

E. I have no idea

```
def onKeyPress(app, key):
    app.text += key
    return key
```

Animation

Examples:

- Bouncing DVD Logo
- Pong

Debugging: Pause Animation

Even more debugging options with graphics

(but animations can make it more challenging)

```
def onAppStart(app):
    app.paused = True

def onStep(app):
    if not app.paused:
        takeStep(app)

def takeStep(app):
    # Update app for one step
    pass
```

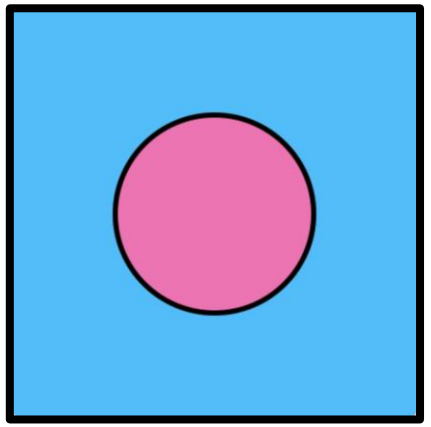
```
def onKeyPress(app, key):
    if key == 's' and app.paused:
        takeStep(app)
    elif key == 'p':
        # Flip paused stats
        app.paused = not app.paused
```

Debugging: Pause Animation

Even more debugging options with graphics

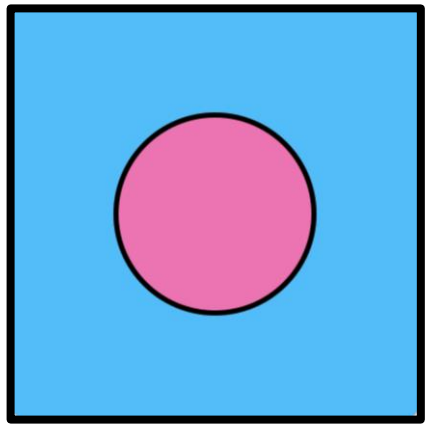
(but animations can make it more challenging)

```
def onAppStart(app):
    app.ballX = app.width/2
    app.ballY = app.height/2
    app.ballR = 0.25*app.width

    app.dX = app.ballR

def takeStep(app):
    app.ballX += app.dX

def reDrawAll(app):
    drawCircle(app.ballX, app.ballY,
        app.ballR, fill=app.ballColor,
        border='black', borderWidth=5)
```



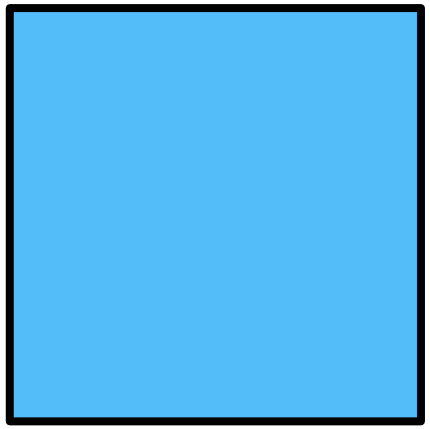
Poll 8

```
def onAppStart(app):
    ...
    app.dX = app.ballR

def onKeyPress(app, key):
    app.ballX += app.dX

def reDrawAll(app):
    drawCircle(app.ballX, app.ballY, app.ballR)
```

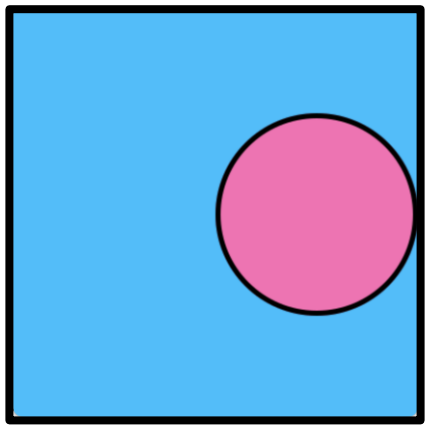


What happens after we press ‘s’?

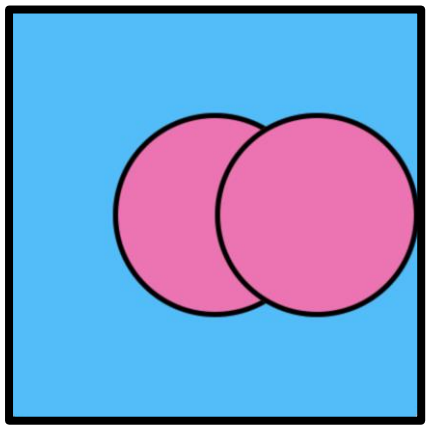
A)



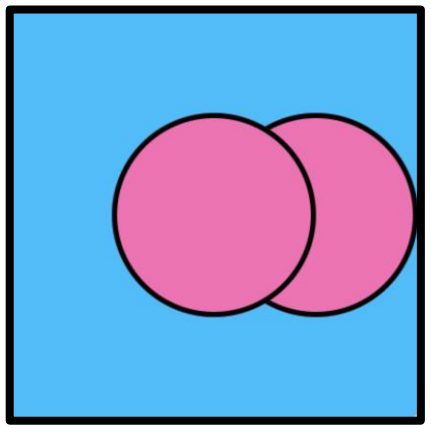
B)

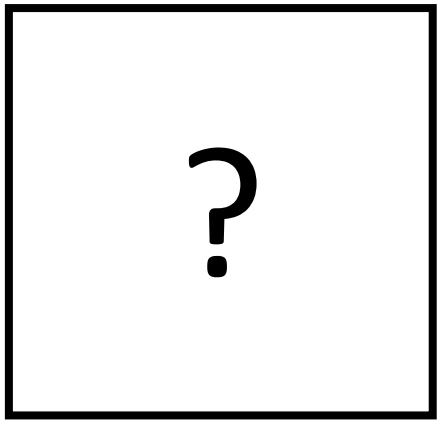


C)



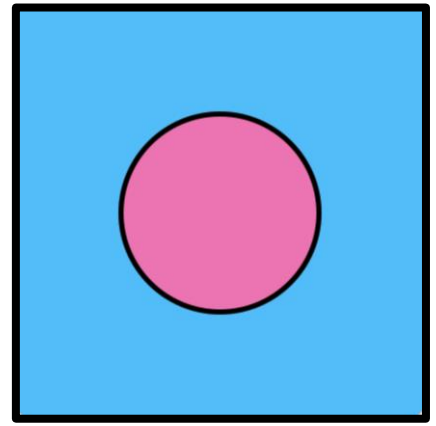
D)

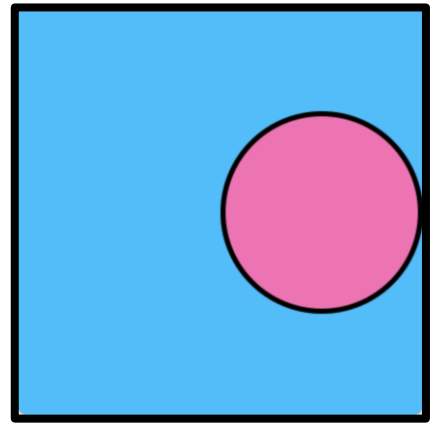




Poll 9

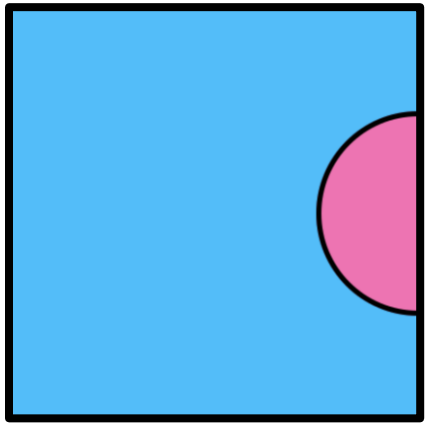
```
def onAppStart(app):
    ...
    app.dX = app.ballR

def takeStep(app):
    app.ballX += app.dX

def reDrawAll(app):
    drawCircle(app.ballX, app.ballY, app.ballR)
```



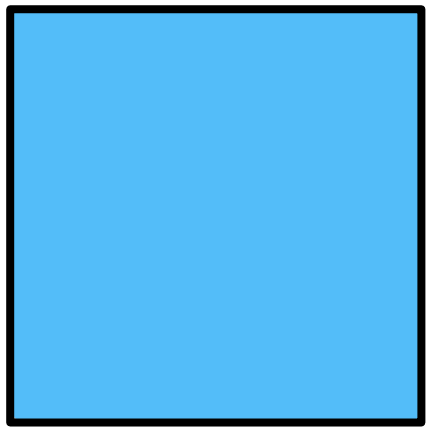


After pressing ‘s’ once, what happens after we press ‘s’ again?

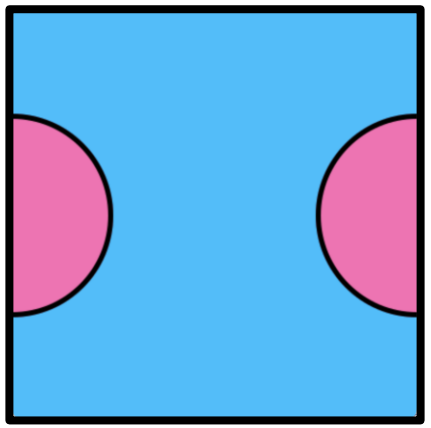
A)



B)



C)



D)

crash!

?

App Skeleton

Tip: Super convenient for sandbox

Note: Next quiz we'll provide function names

appSkeleton.py

```python
from cmu_graphics import *

### Controller

def onAppStart(app):
    app.paused = True
    # app.stepsPerSecond = 30

def onStep(app):
    if not app.paused:
        takeStep(app)

def takeStep(app):
    pass

def onKeyPress(app, key):
    if key == 's' and not app.paused:
        takeStep(app)

    elif key == 'p':
        app.paused = not app.paused

def onKeyRelease(app, key):
    pass

def onKeyHold(app, key):
    pass

def onMousePress(app, mouseX, mouseY):
    pass

def onMouseRelease(app, mouseX, mouseY):
    pass

def onMouseDrag(app, mou
    pass

def onMouseMove(app, mou
    pass

def onResize(app):
    pass

### View

def redrawAll(app):
    pass

### Main

def main():
    runApp()

main()
```