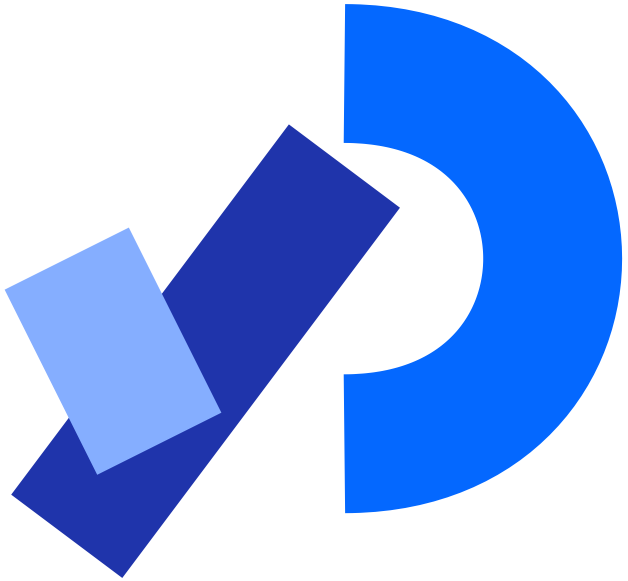


Processing

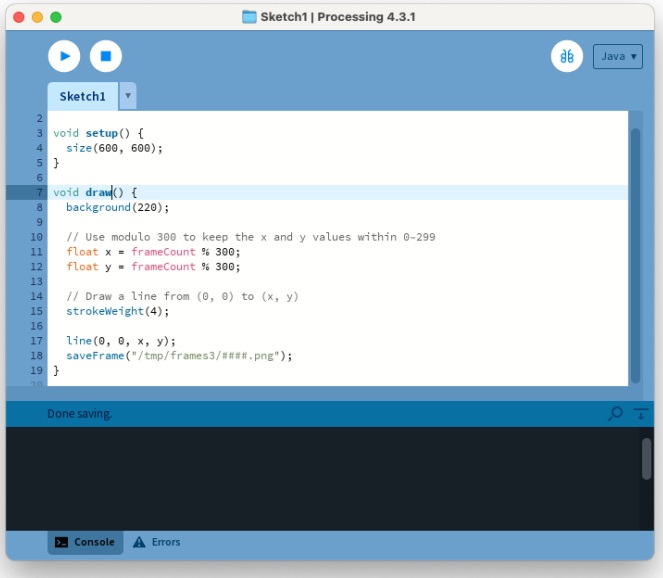
What is Processing?

- *Processing* is a free graphics library built for the electronic arts and visual design communities

- *Processing* is not a programming language but an **arts-centric system for learning, teaching, and making visual form with code**

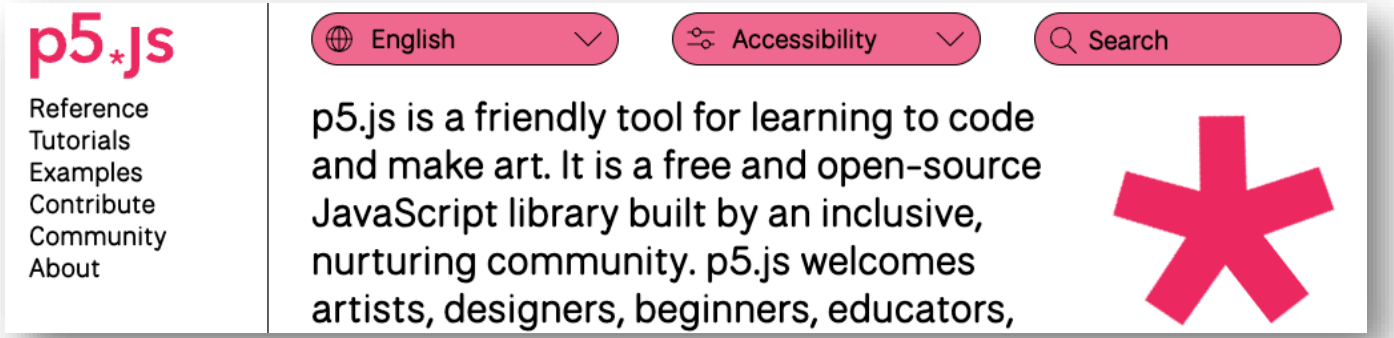
- Allows the developer to quickly create interactive graphics-based programs

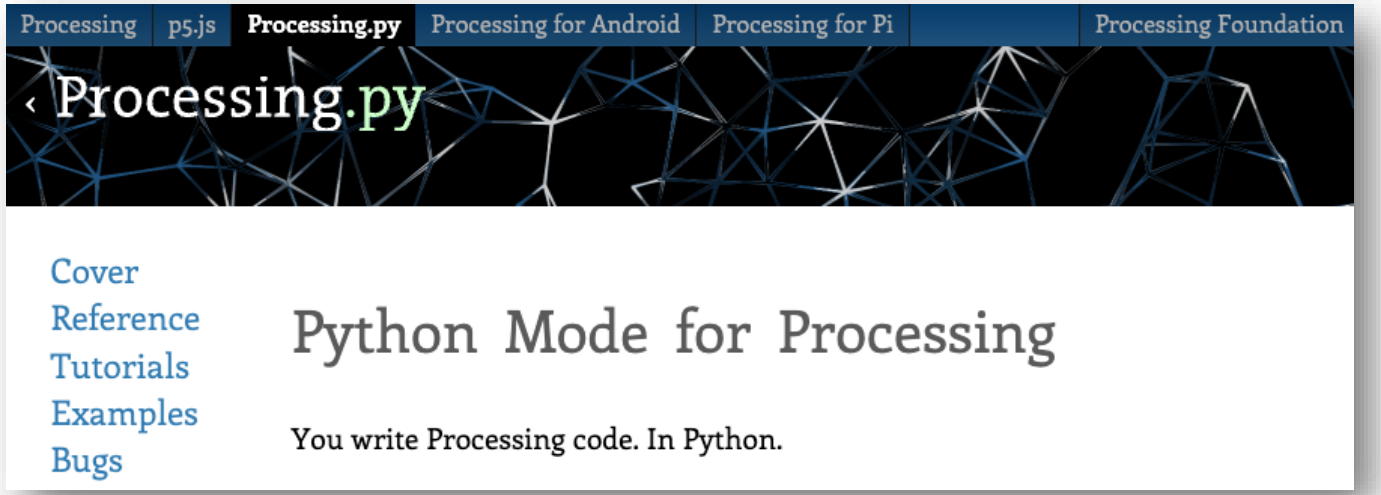
Processing IDE



What is Processing?

- *Processing* has expanded from its Java roots





What is Processing?

Structure

(comment)
"" "" (multiline comment)
() (parentheses)
, (comma)
. (dot)
: (Colon)
= (assign)
False
Globals
None
Slice
String Formatting
True
[] (Index brackets)
add_library
class
def
del
draw()
except
exit()
in
len()

Shape

createShape()

2D Primitives

arc()
circle()
ellipse()
line()
point()
quad()
rect()
square()
triangle()

Curves

bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()

Color

Setting

background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

Creating & Reading

alpha()
blendColor()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

Image

Open Processing

- *Processing* has a large online community of programmers who share their processing sketches

https://openprocessing.org/

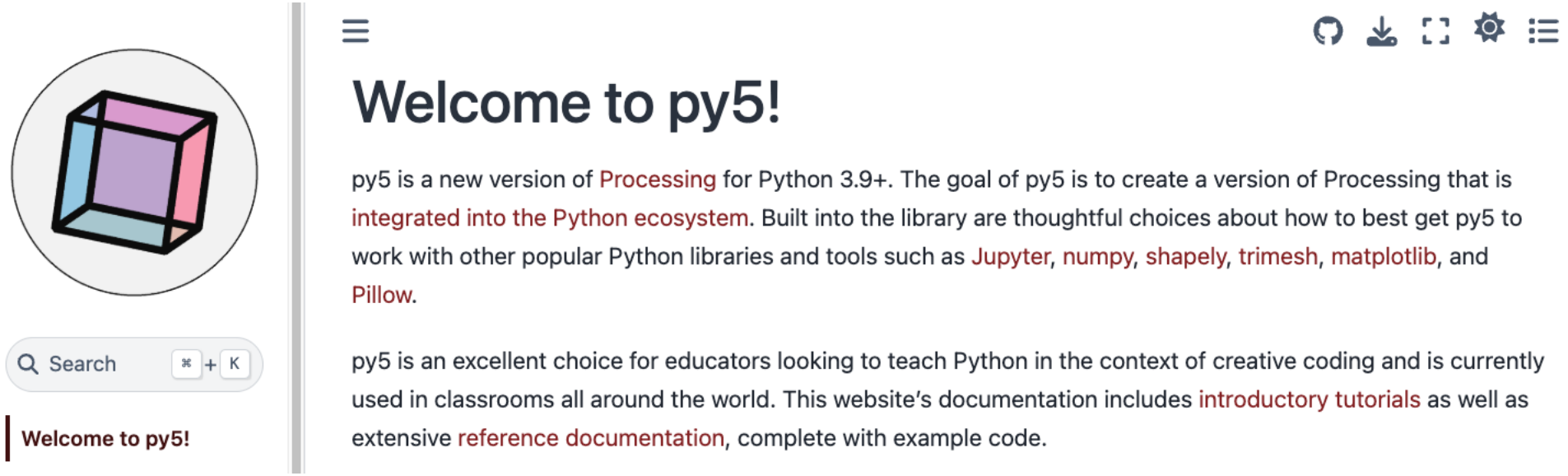
History of Proce55ing aka. p5

- *Processing* was initiated in 2001 by Casey Reas and Ben Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab

- Originally, *Processing* had used the domain **proce55ing.net**, because the processing domain was taken

- The abbreviated term **p5** is still occasionally used (e.g. in "p5.js" or "**py5**") in reference to the old domain name

Integrating Processing and Jupyter Notebooks



https://py5coding.org/

https://py5coding.org/reference/summary.html

py5 Reference Summary

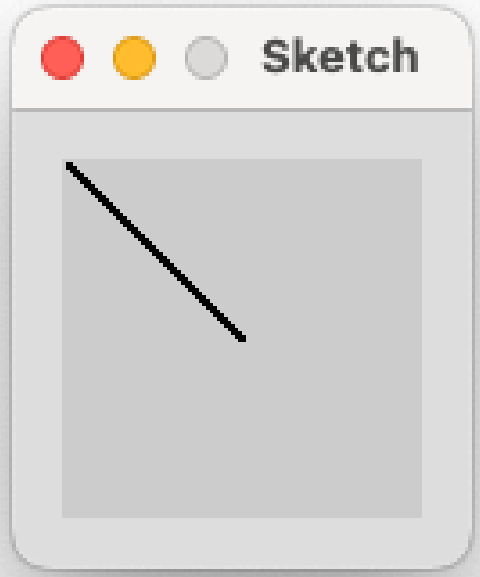
Drawing Shapes

Basic Elements

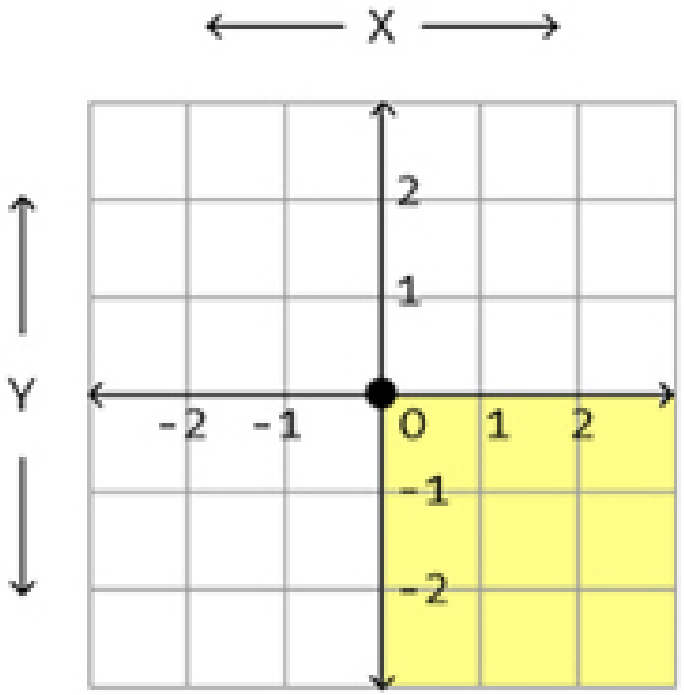
- arc() - Draws an arc to the screen.
- circle() - Draws a circle to the screen.
- ellipse() - Draws an ellipse (oval) - to the screen.
- ellipse_mode() - Modifies the location from which ellipses and circles are drawn by changing the way in which values given are interpreted.
- line() - Draws a line (a direct path between two points) - to the screen.
- lines() - Draw a collection of lines to the screen.
- point() - Draws a point, a coordinate in space at the dimension of one pixel.
- points() - Draw a collection of points, each a coordinate in space at the dimension of one pixel.
- quad() - A quad is a quadrilateral, a four sided polygon.
- rect() - Draws a rectangle to the screen.
- rect_mode() - Modifies the location from which rectangles and squares are drawn by changing the way in which values given are interpreted.
- square() - Draws a square to the screen.
- triangle() - A triangle is a plane created by connecting three points.

Hello Processing

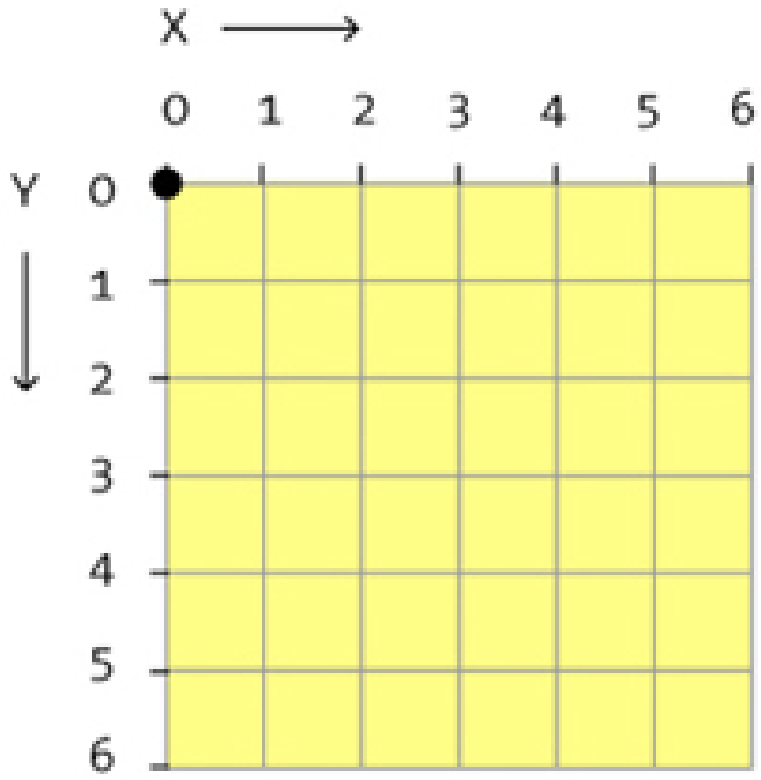
```
# The most basic processing sketch.
# aka. "Hello World!" in processing.

import py5

def draw():
    py5.line(1, 1, 50, 50)

py5.run_sketch()
```



Processing Coordinates



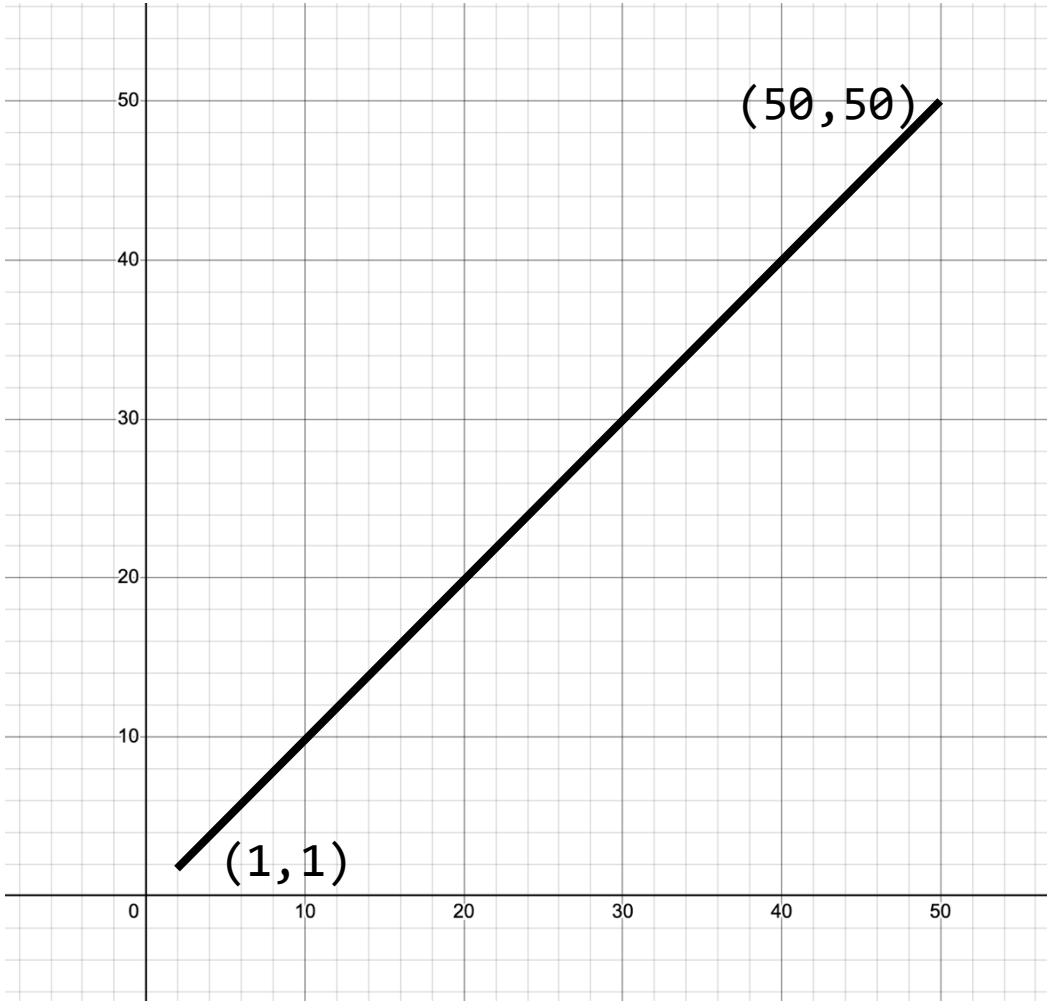
Cartesian coordinate system



***Processing* coordinate system**

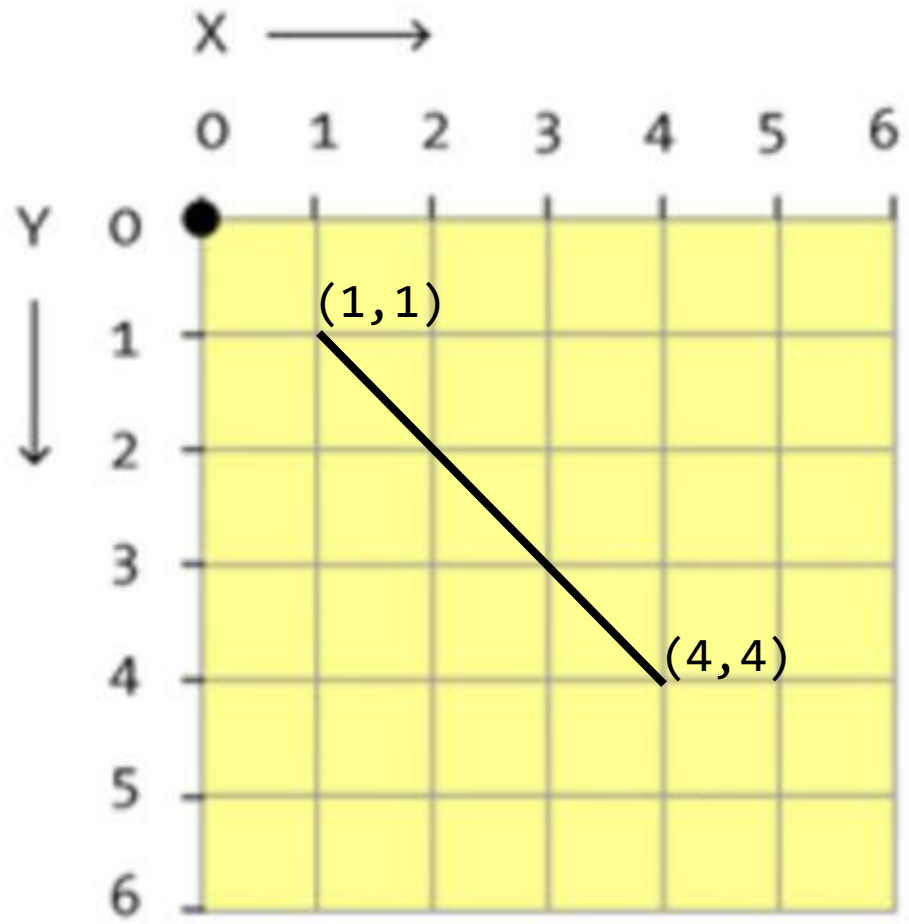
```
py5.line(1, 1, 50, 50)
```

Draw a line from **(1, 1)** to **(50, 50)**



```
py5.line(1, 1, 4, 4)
```

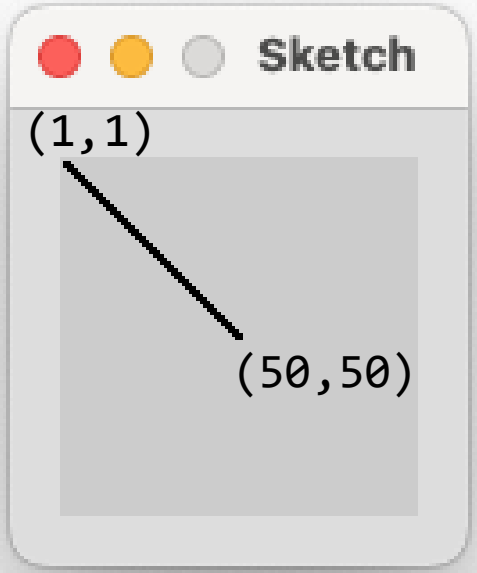
Draw a line from **(1, 1)** to **(4, 4)**



Processing Coordinates

```
py5.line(1, 1, 50, 50)
```

Draw a line from **(1, 1)** to **(50, 50)**



Hello Processing

```
# The setup function where you configure your sketch.
def setup():
    py5.size(600, 600)

# The draw function where you draw your sketch.
def draw():
    py5.line(1, 1, 300, 300)

py5.run_sketch()
```

Hello Processing

```
# The setup function where you configure your sketch.
def setup():
    py5.size(600, 600)

# The draw function where you draw your sketch.
def draw():
    py5.line(1, 1, 300, 300)

py5.run_sketch()
```

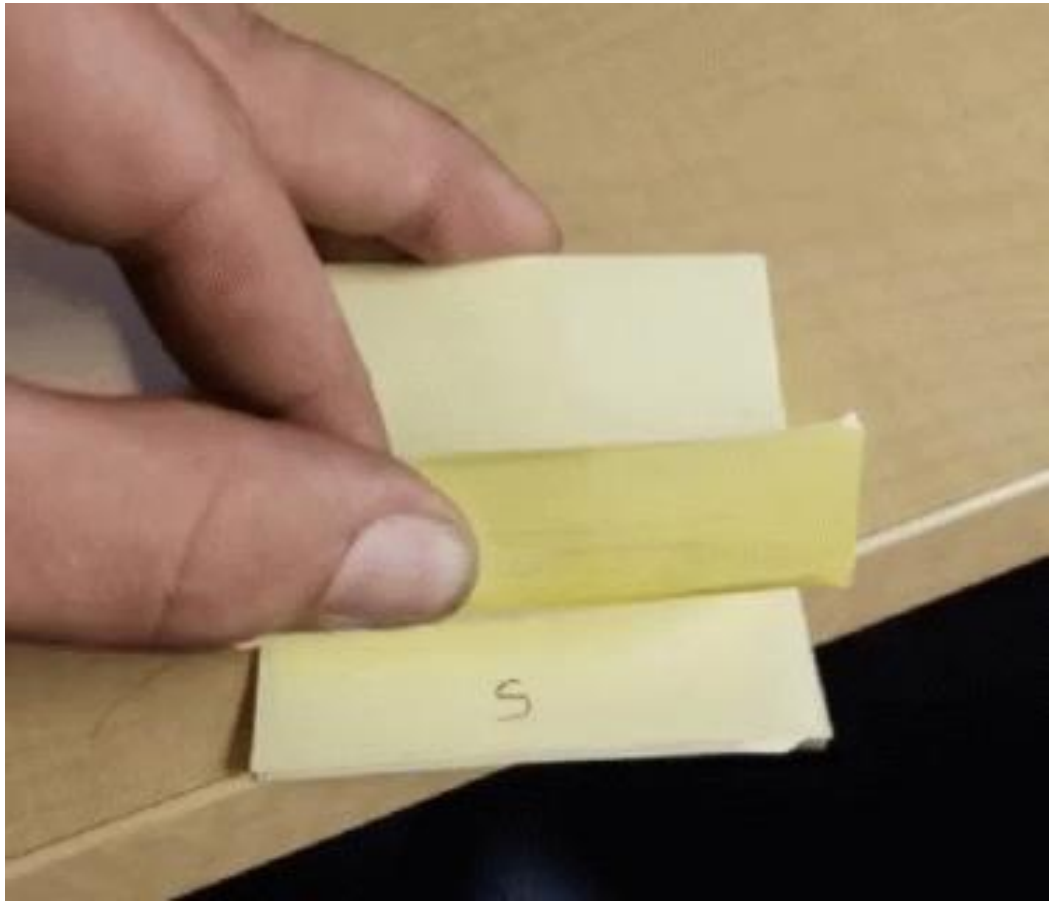
The setup function is called only one time

Hello Processing

```
# The setup function where you configure your sketch.
def setup():
    py5.size(600, 600)

# The draw function where you draw your sketch.
def draw():
    py5.line(1, 1, 300, 300)

py5.run_sketch()
```

The draw function is called 60 times per second

Frame Rate Animations



Frame Rate Animations

```
# The draw function is called once every 60 seconds.
def draw():
    # When animating you must redraw the background.
    py5.background(200)
    py5.text_size(32)
    py5.fill(5)
    # The py5.frame_count is a count of the number of
    # frames that have been shown.
    py5.text(py5.frame_count, 10, 50)

py5.run_sketch()
```

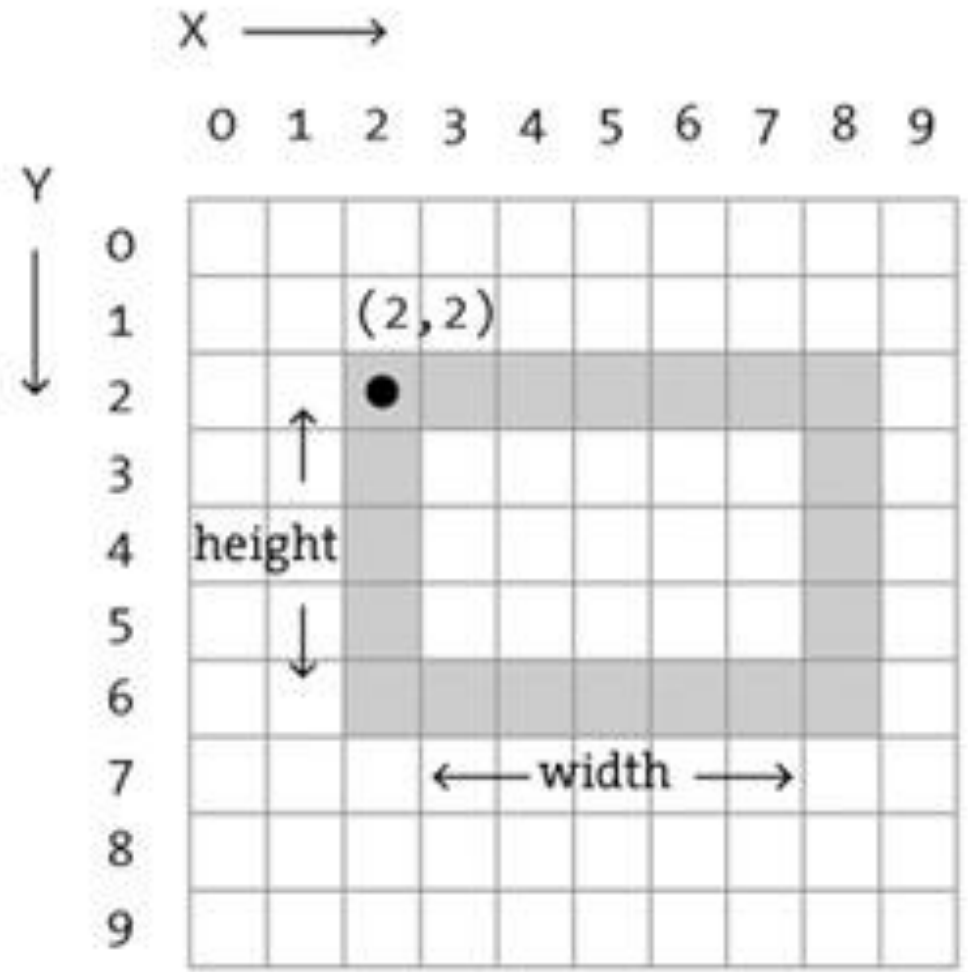
Frame Rate Animations

```
def draw():
    # Gray background
    py5.background(220)

    x = py5.frame_count % 300
    y = py5.frame_count % 300
    py5.line(0, 0, x, y)
```

In-Class Exercise 1

- Image you're an archer shooting an arrow across the screen
- Write a *Processing* sketch that shows an arrow flying from left edge of the canvas to the right edge:

Rectangle



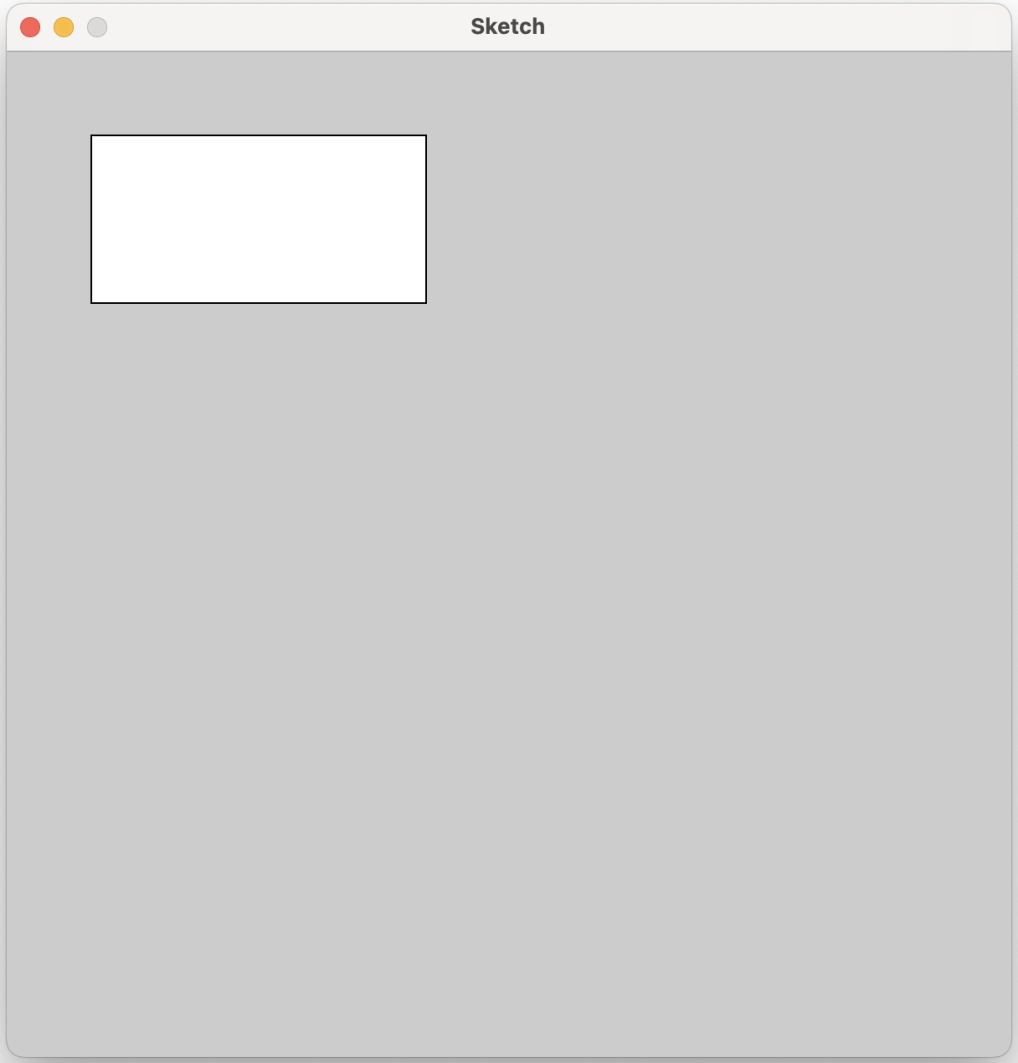

```
rect(x,y,width,height)
```

Example:

```
rect(2,2,7,5)
```

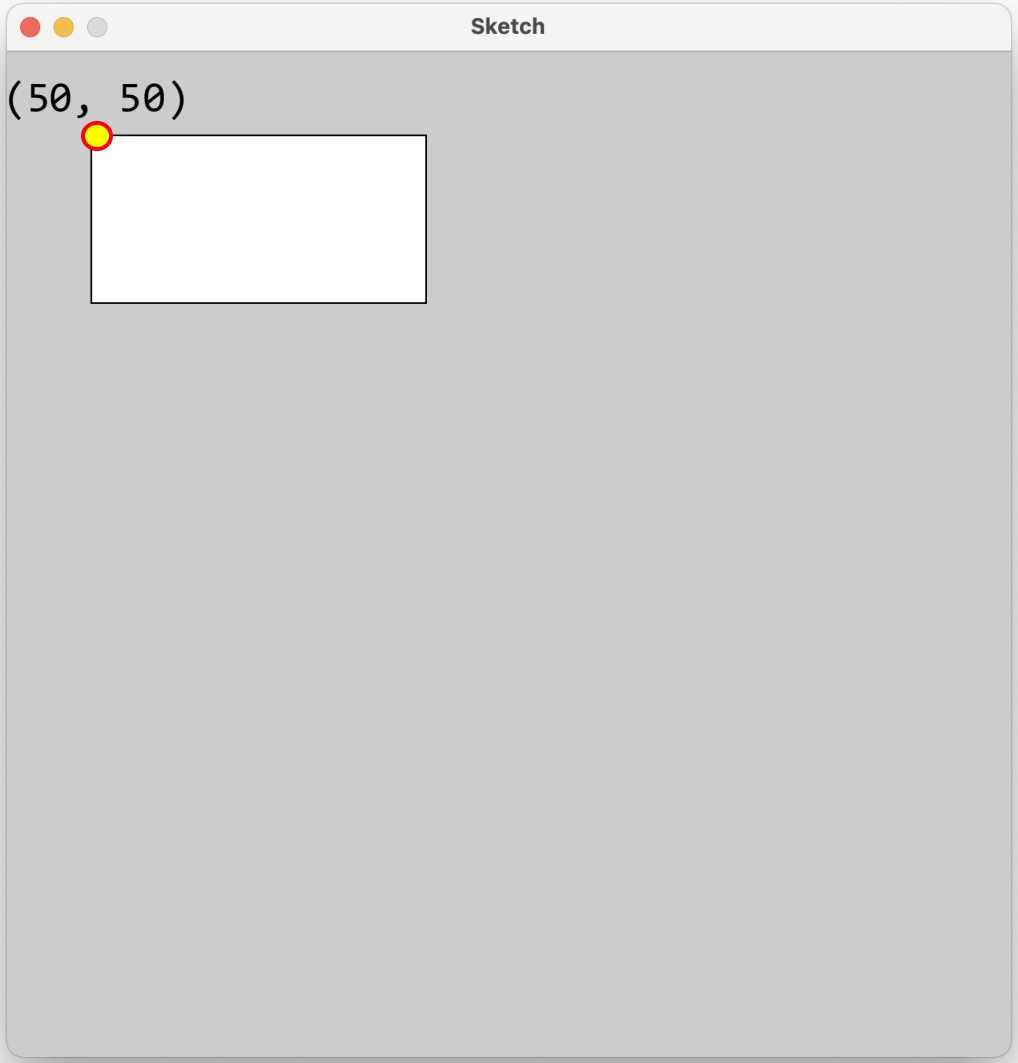
Rectangle

```
def draw():
    # Starting at point (50, 50)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect(50, 50, 200, 100)
```

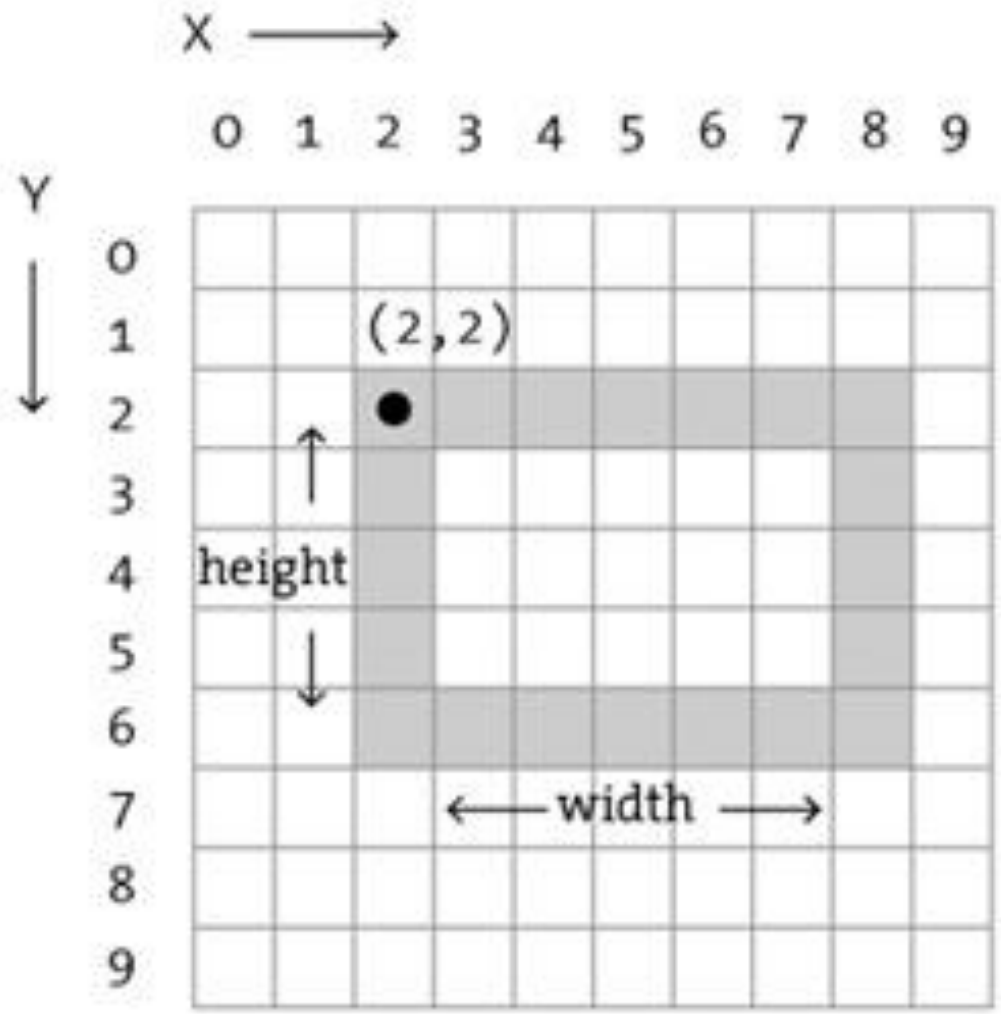


Rectangle

```
def draw():
    # Starting at point (50, 50)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect(50, 50, 200, 100)
```



Rectangle Modes: **Corner**




```
# Default rectangle mode
py5.rect_mode(py5.CORNER)
```

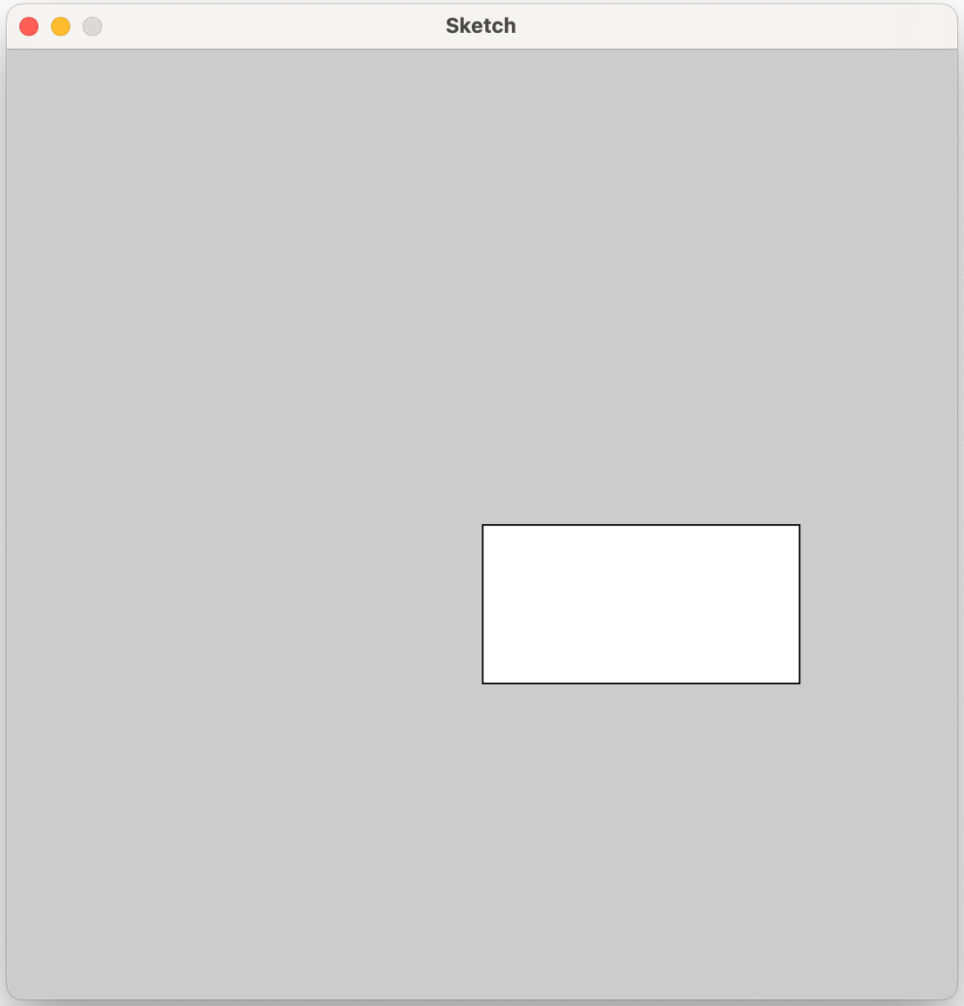
```
rect(x,y,width,height)
```

Example:

```
rect(2,2,7,5)
```

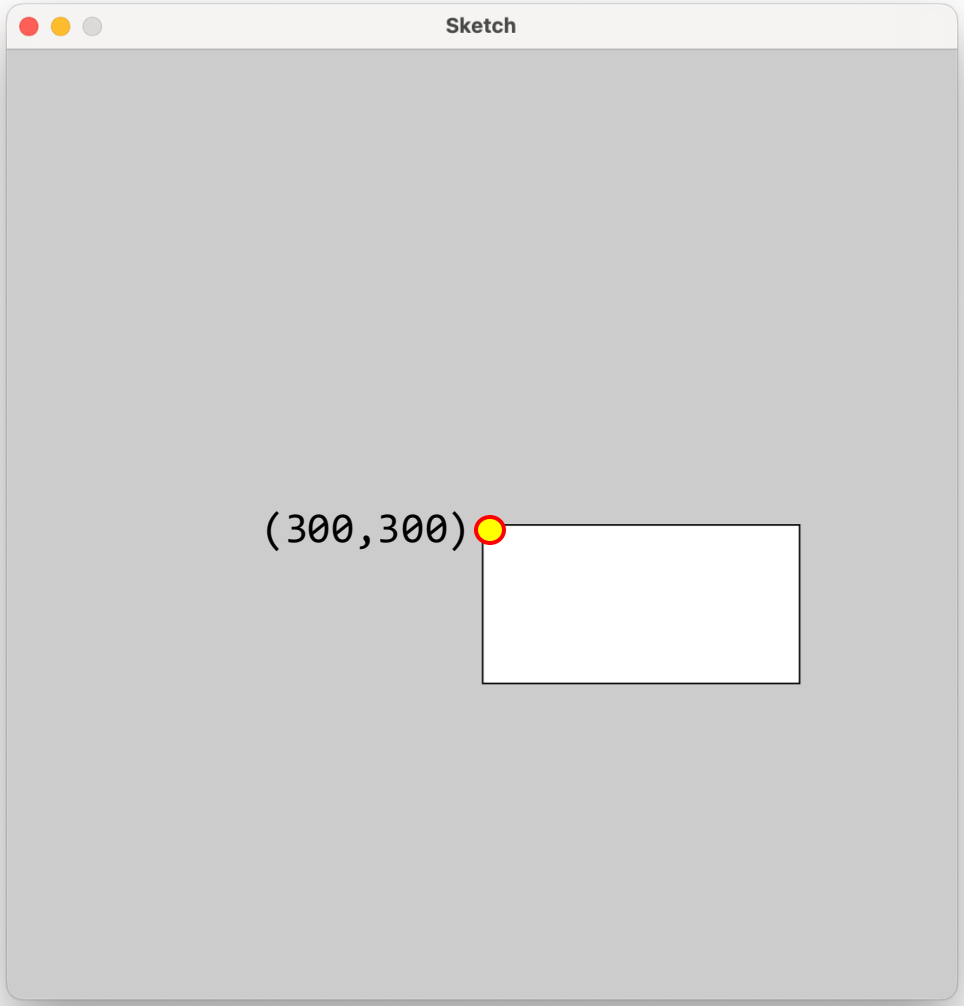
Rectangle Modes: **Corner**

```
def draw():
    # Starting at point (300, 300)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect_mode(py5.CORNER)
    py5.rect(300, 300, 200, 100)
```

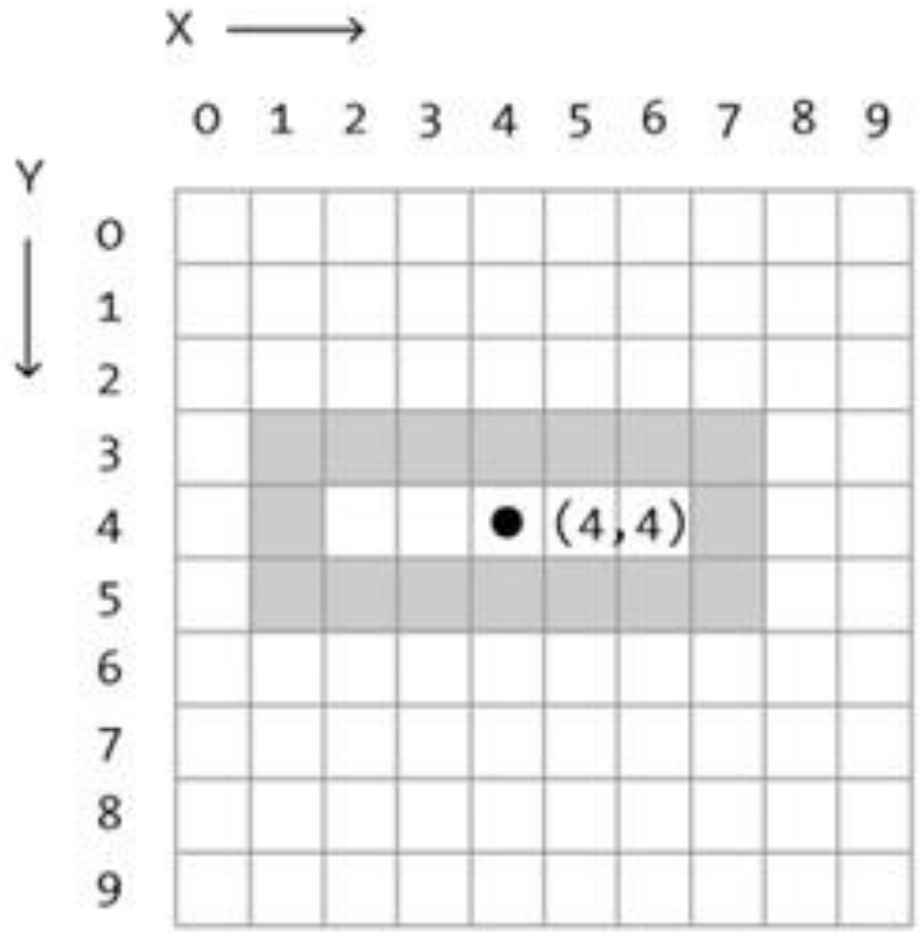


Rectangle Modes: **Corner**

```
def draw():
    # Starting at point (300, 300)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect_mode(py5.CORNER)
    py5.rect(300, 300, 200, 100)
```



Rectangle Modes: **Center**



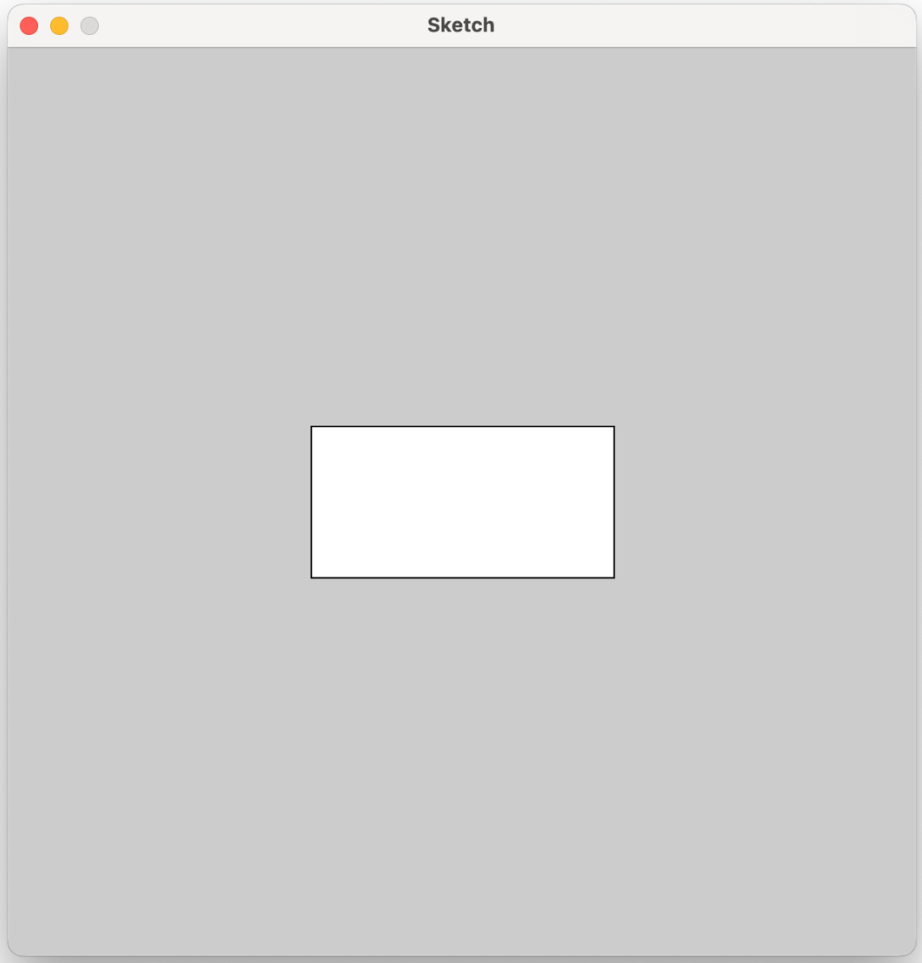

```
rectMode(CENTER)
rect(x,y,width,height)
```

Example:

```
rectMode(CENTER)
rect(4,4,7,3)
```

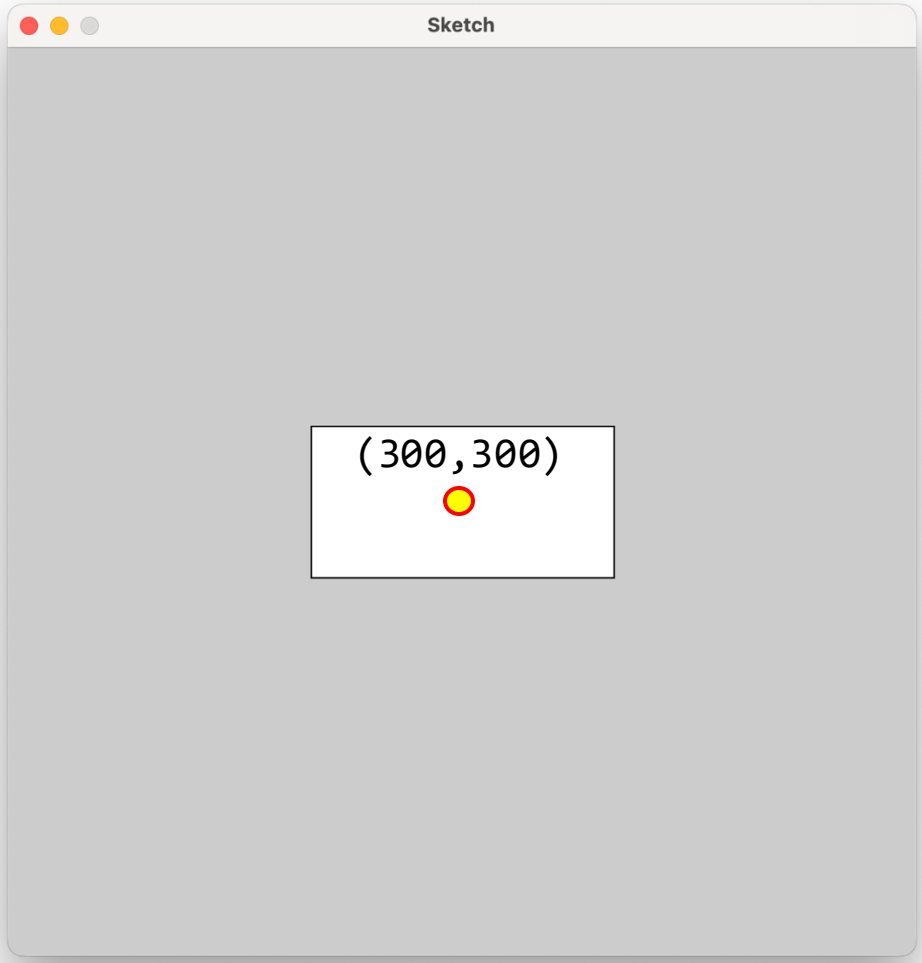
Rectangle Modes: **Center**

```
def draw():
    # Starting at point (300, 300)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect_mode(py5.CENTER)
    py5.rect(300, 300, 200, 100)
```



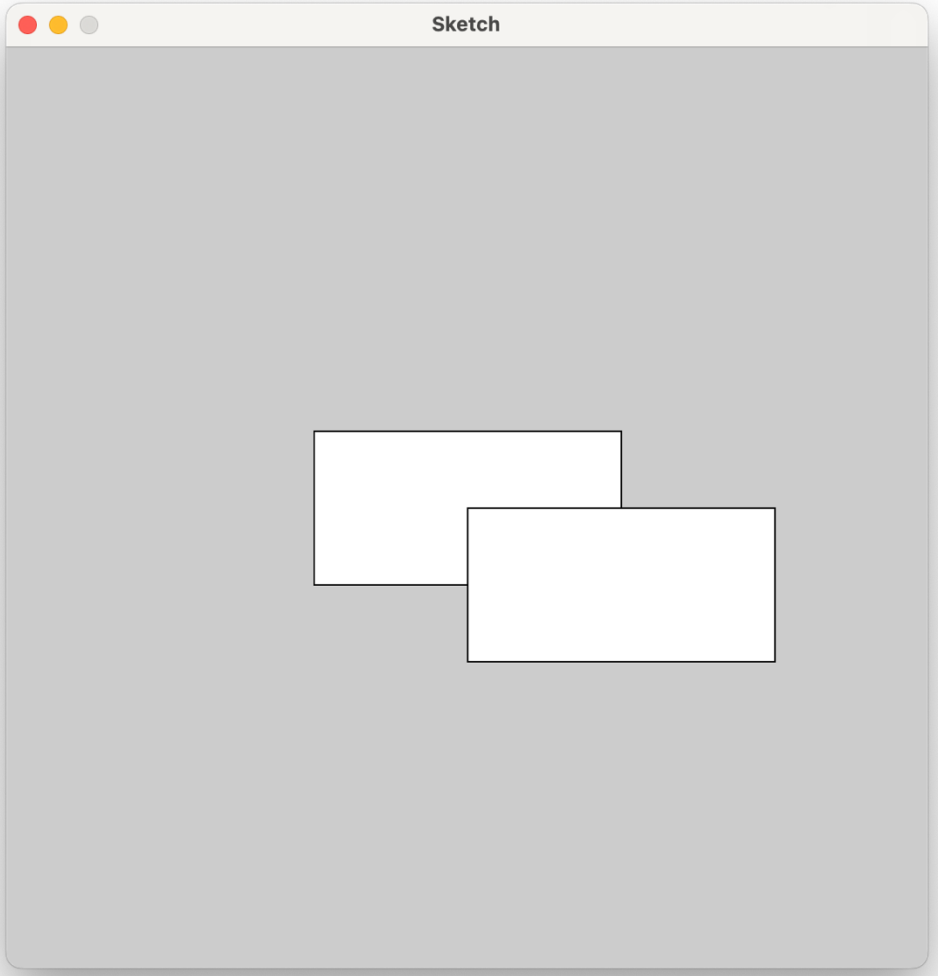
Rectangle Modes: **Center**

```
def draw():
    # Starting at point (300, 300)
    # Draw a rectangle with a
    # width 200 and height 100
    py5.rect_mode(py5.CENTER)
    py5.rect(300, 300, 200, 100)
```



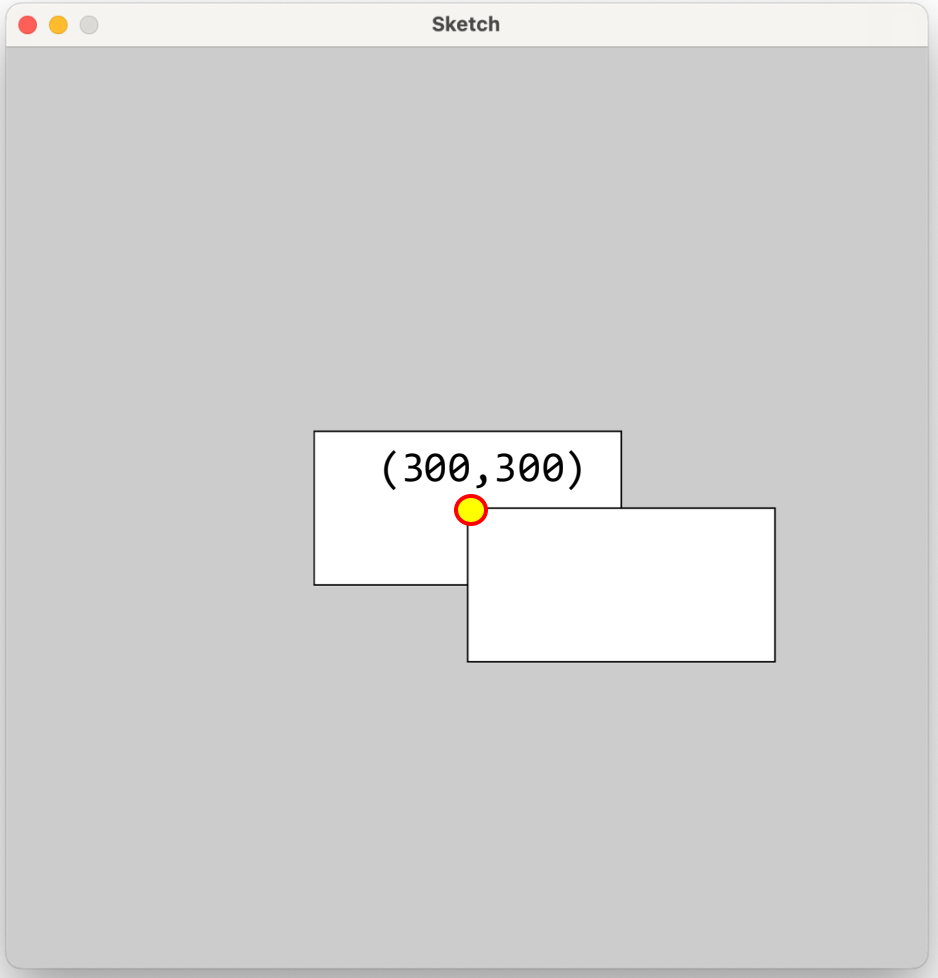
Rectangle Modes

```
def draw():
    py5.rect_mode(py5.CENTER)
    py5.rect(300, 300, 200, 100)
    py5.rect_mode(py5.CORNER)
    py5.rect(300, 300, 200, 100)
```

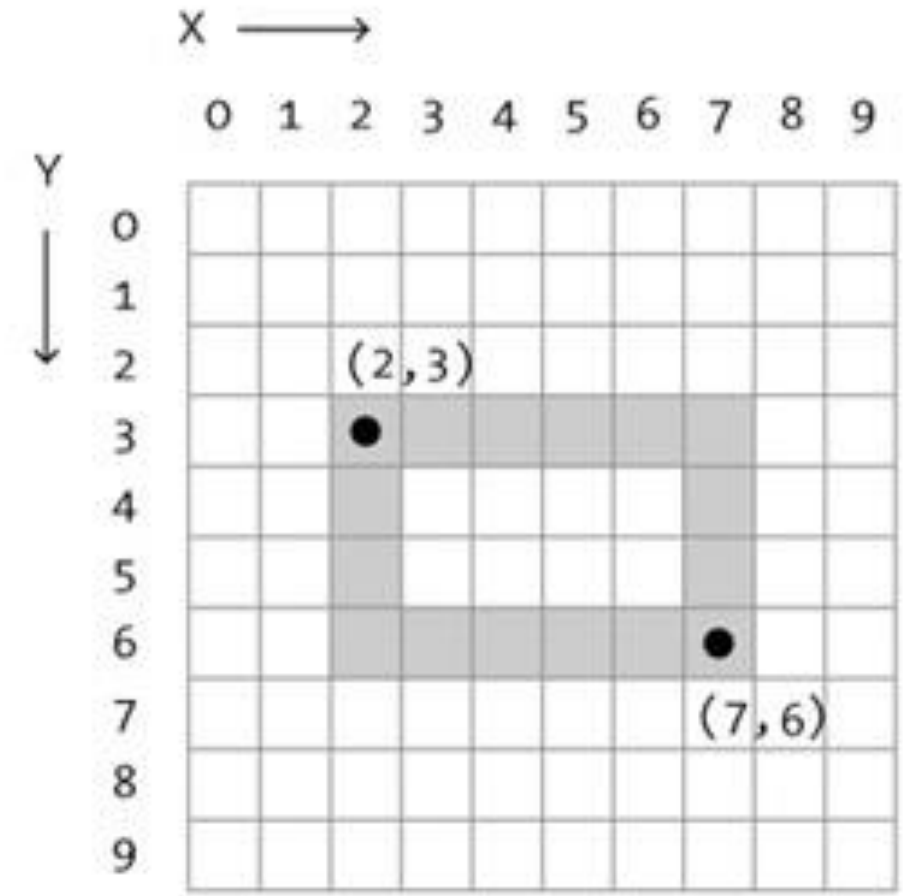


Rectangle Modes

```
def draw():
    py5.rect_mode(py5.CENTER)
    py5.rect(300, 300, 200, 100)
    py5.rect_mode(py5.CORNER)
    py5.rect(300, 300, 200, 100)
```



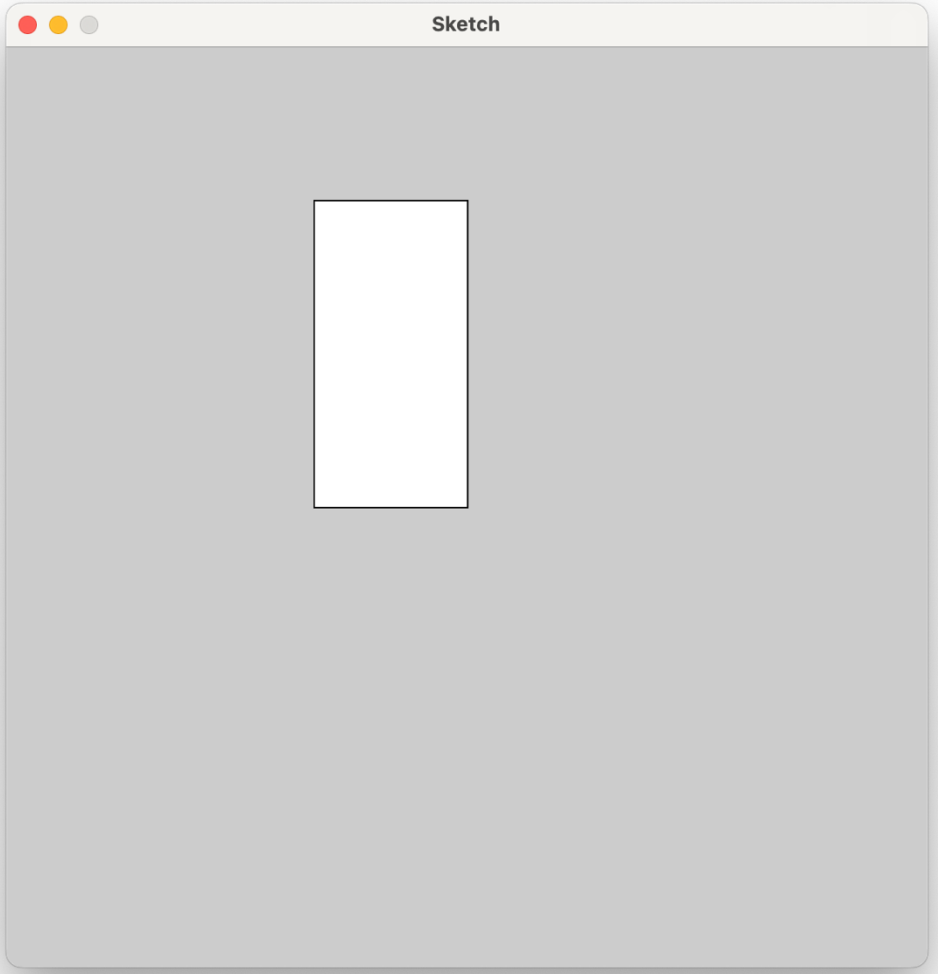
Rectangle Modes: **Corners**




```
rectMode(CORNERS)
rect(x1,y1,x2,y2)

Example:
rectMode(CORNERS)
rect(2,3,7,6)
```

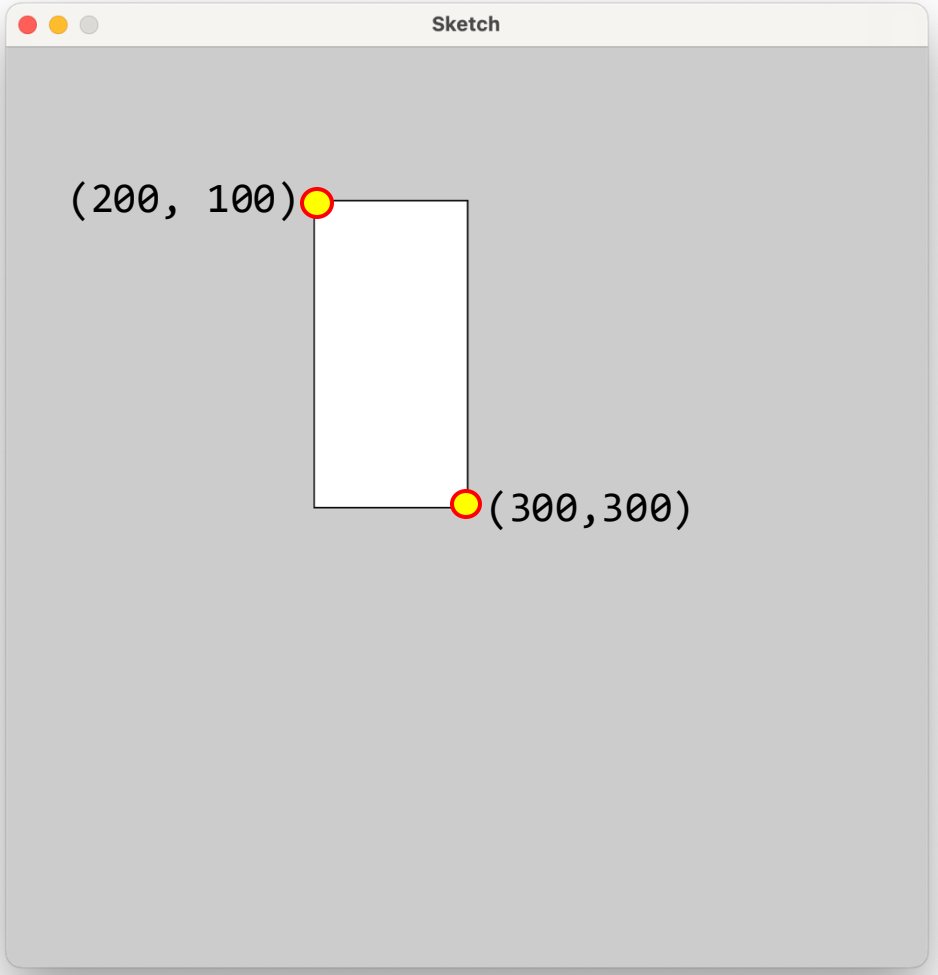
Rectangle Modes: **Corners**

```
def draw():
    py5.rect_mode(py5.CORNERS)
    py5.rect(300, 300, 200, 100)
```



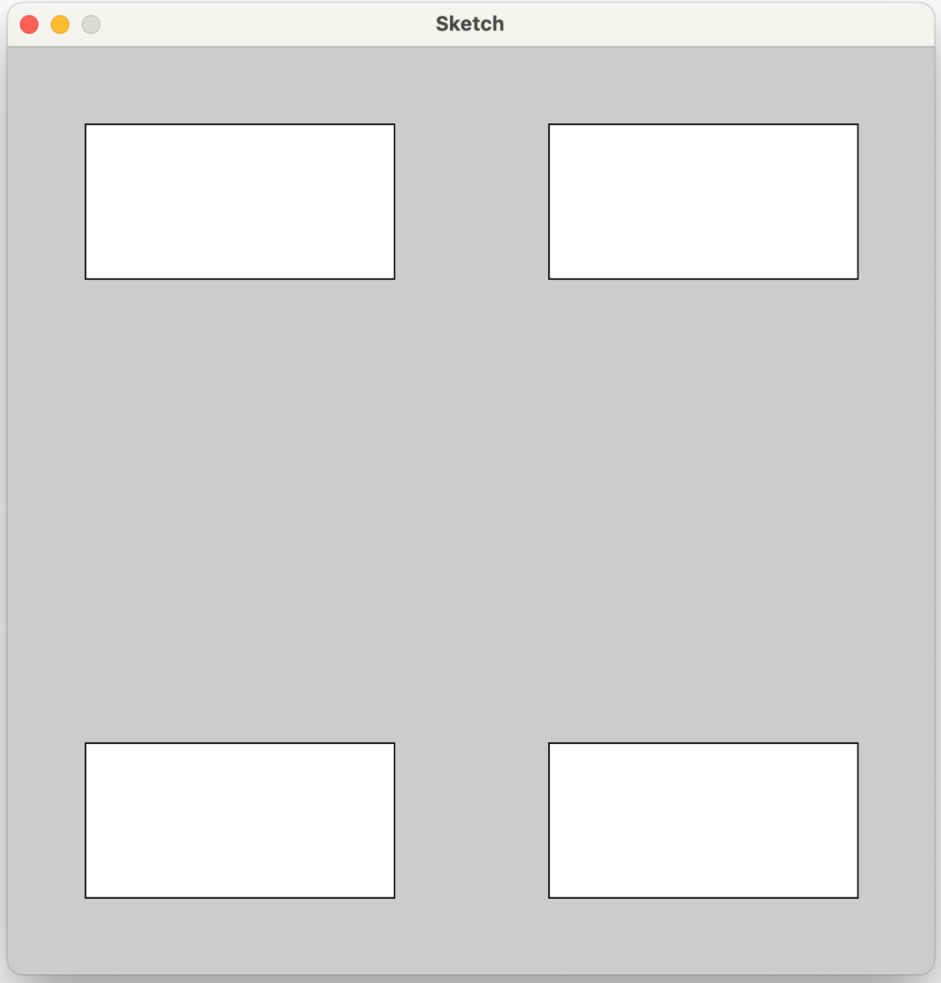
Rectangle Modes: **Corners**

```
def draw():
    py5.rect_mode(py5.CORNERS)
    py5.rect(300, 300, 200, 100)
```



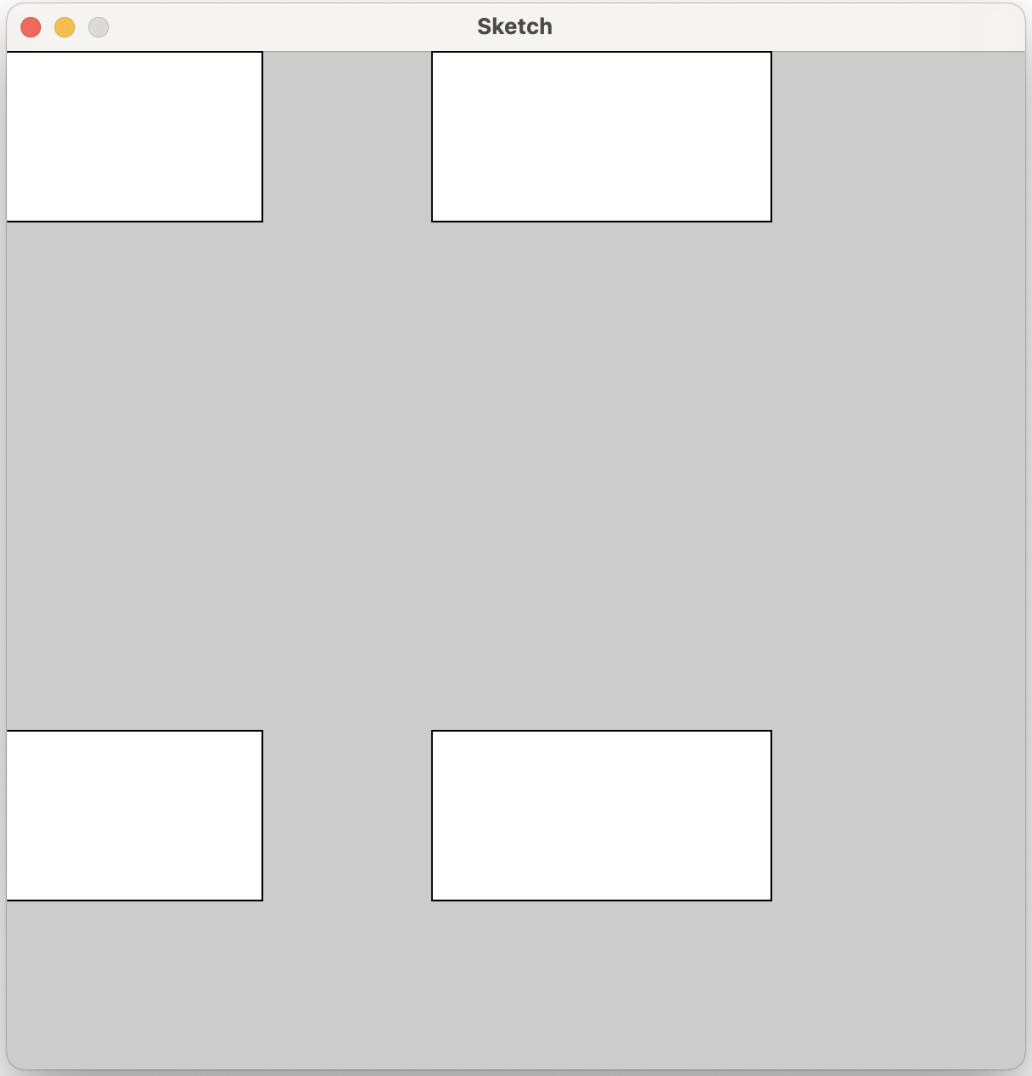
Rectangle Modes

```
def draw():
    py5.rect(50, 50, 200, 100)
    py5.rect(350, 50, 200, 100)
    py5.rect(50, 450, 200, 100)
    py5.rect(350, 450, 200, 100)
```



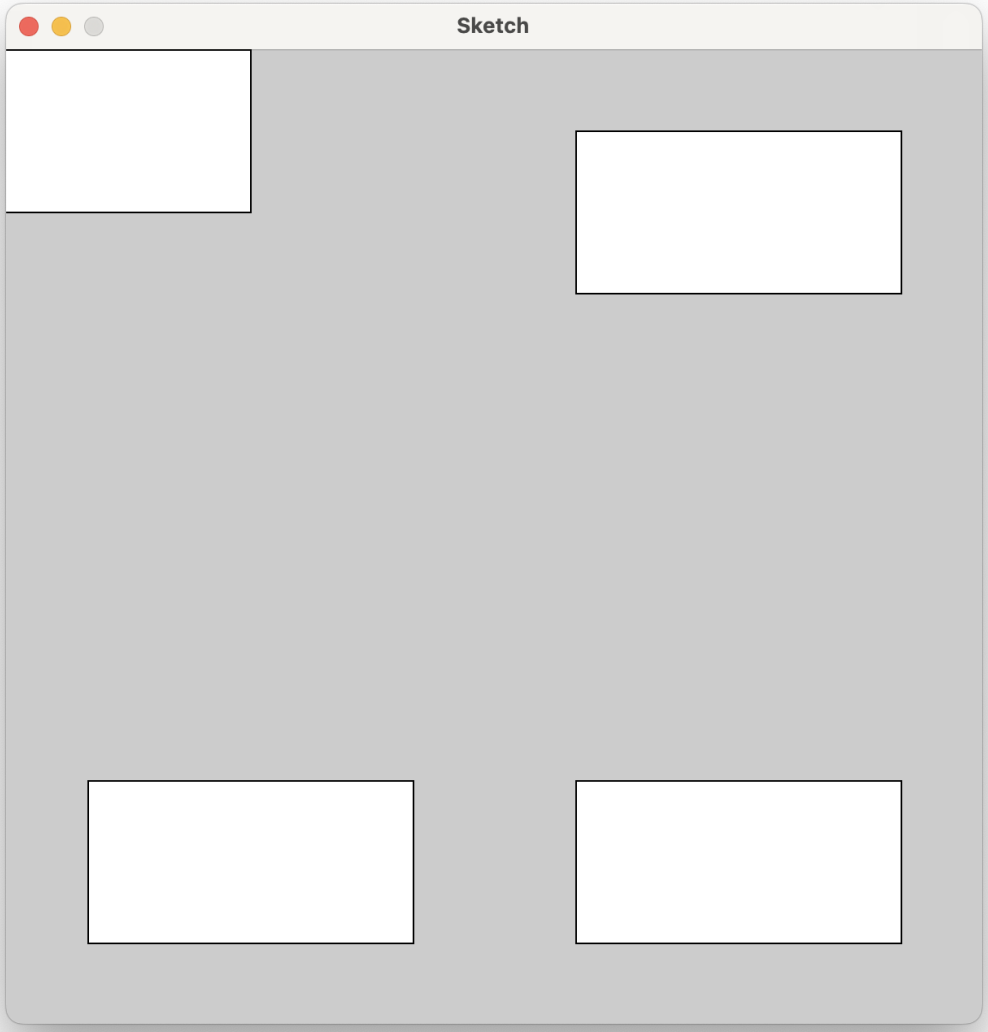
Rectangle Modes

```
def draw():
    # Will modify all rectangles
    py5.rect_mode(py5.CENTER)
    py5.rect(50, 50, 200, 100)
    py5.rect(350, 50, 200, 100)
    py5.rect(50, 450, 200, 100)
    py5.rect(350, 450, 200, 100)
```



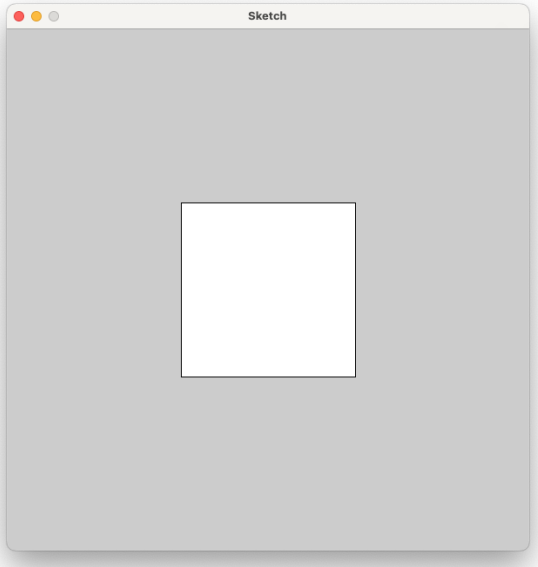
Saving and Restoring Sketch State

```
def draw():
    py5.push()  # Save state

    py5.rect_mode(py5.CENTER)
    py5.rect(50, 50, 200, 100)

    py5.pop()   # Restore state

    py5.rect(350, 50, 200, 100)
    py5.rect(50, 450, 200, 100)
    py5.rect(350, 450, 200, 100)
```



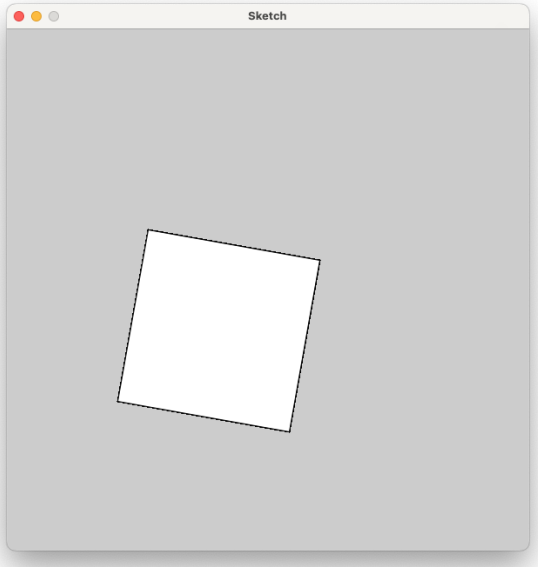
Rotation

```
def draw():
    py5.rect(200, 200, 200, 200)
```



- Angle is provided in radians from 0 to 2π
- Positive numbers rotate clockwise
- Applies to everything that happens afterward

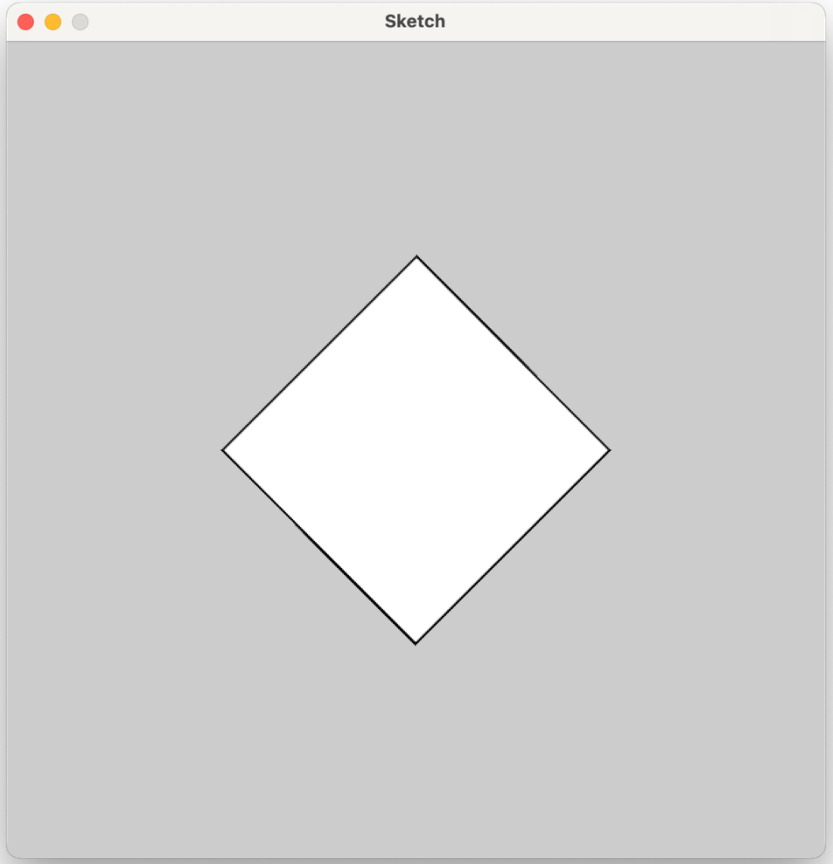
```
def draw():
    py5.rotate(py5.radians(10))
    py5.rect(200, 200, 200, 200)
```



Translation

- Specifies an amount to displace objects within the display window

```
def draw():
    py5.translate(py5.width / 2, py5.height / 2)
    py5.rotate(py5.radians(45))
    py5.rect_mode(py5.CENTER)
    py5.rect(0, 0, 200, 200)
```



In-Class Exercise

- Create a square that ”rolls” from left to right across the canvas