Processing Introduction

CSE 120 Spring 2017

Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

Administrivia

- Assignments:
  - Lightbot Functions due today (4/3)
  - Building a Robot due tomorrow (4/4)
  - Taijitu due Wednesday (4/5)

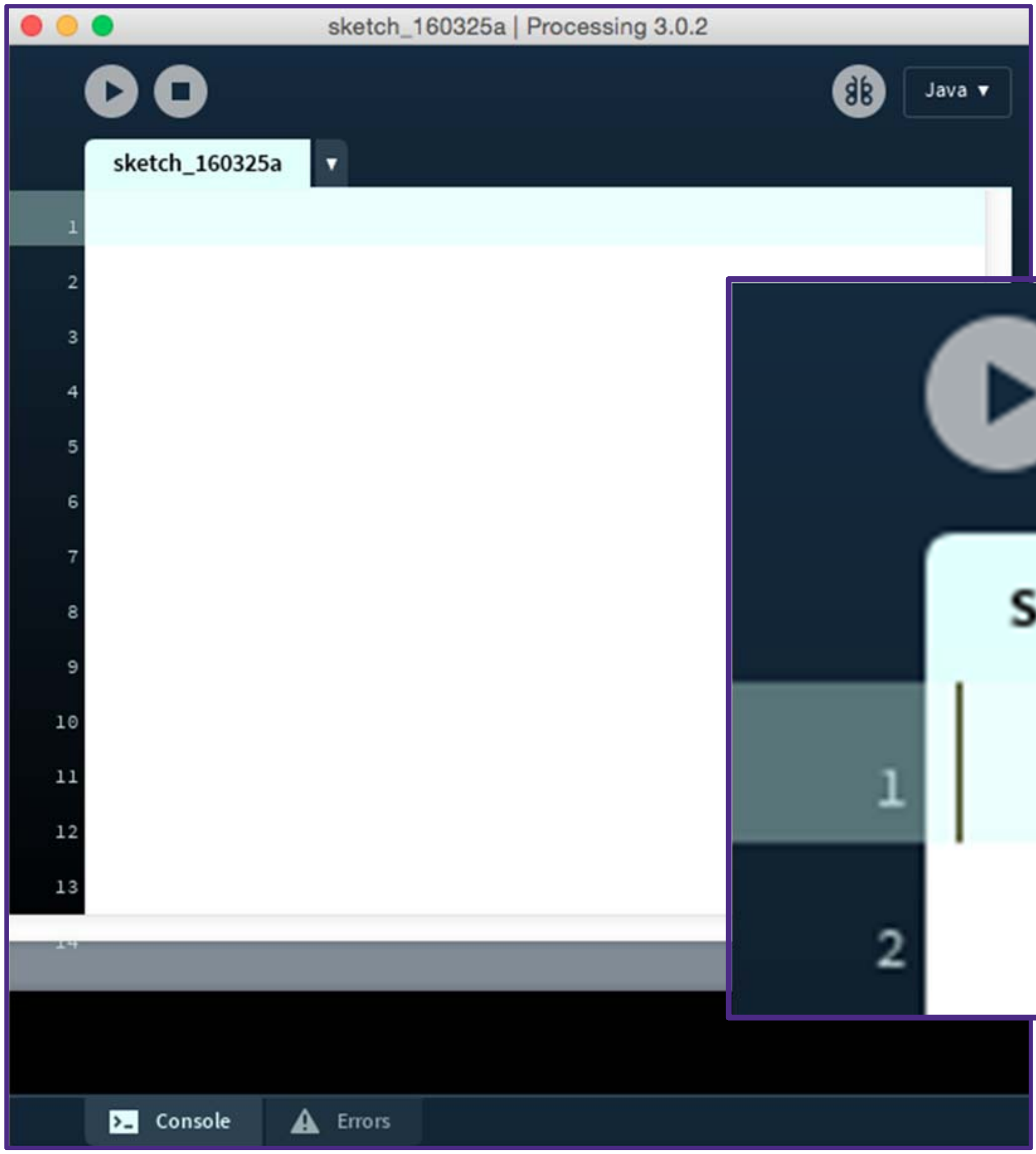
- No “big ideas” lecture this week
  - More time on programming

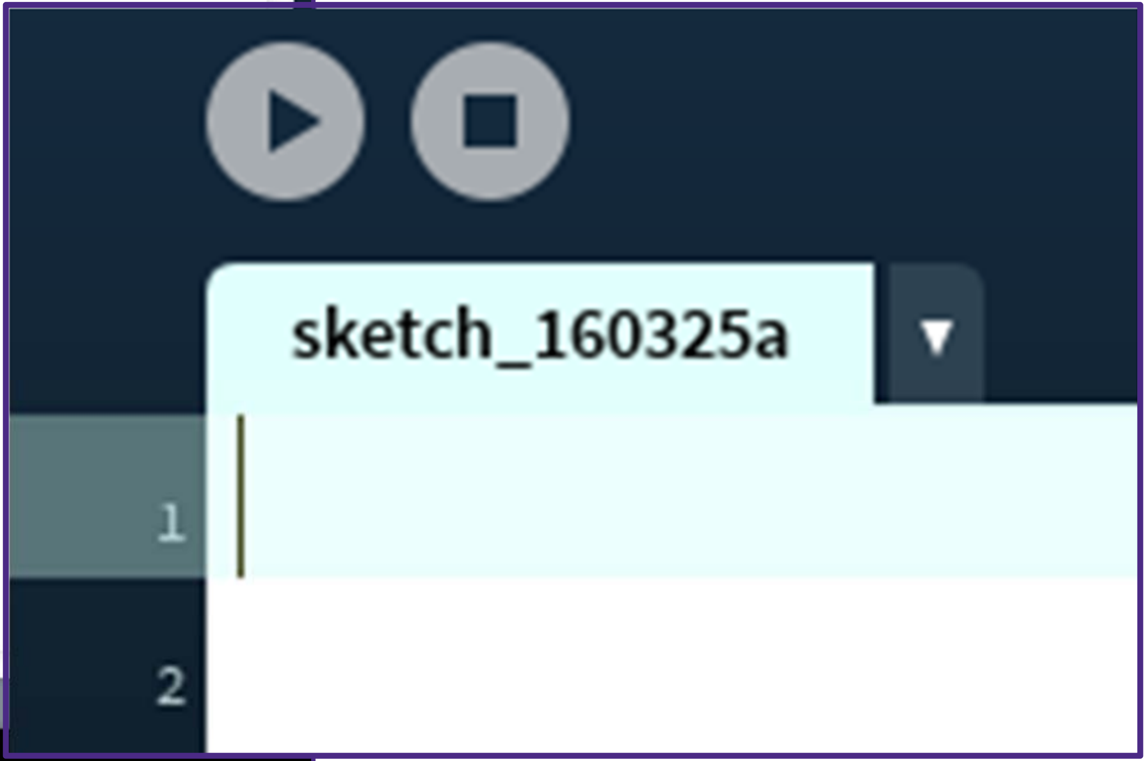
Processing

- Our programming language for this course
  - Text-based language that is good for visuals and interaction
  - Try to focus on ideas and techniques, not the specific commands
  - No language is perfect – Processing has its fair share of quirks and deficiencies ☹

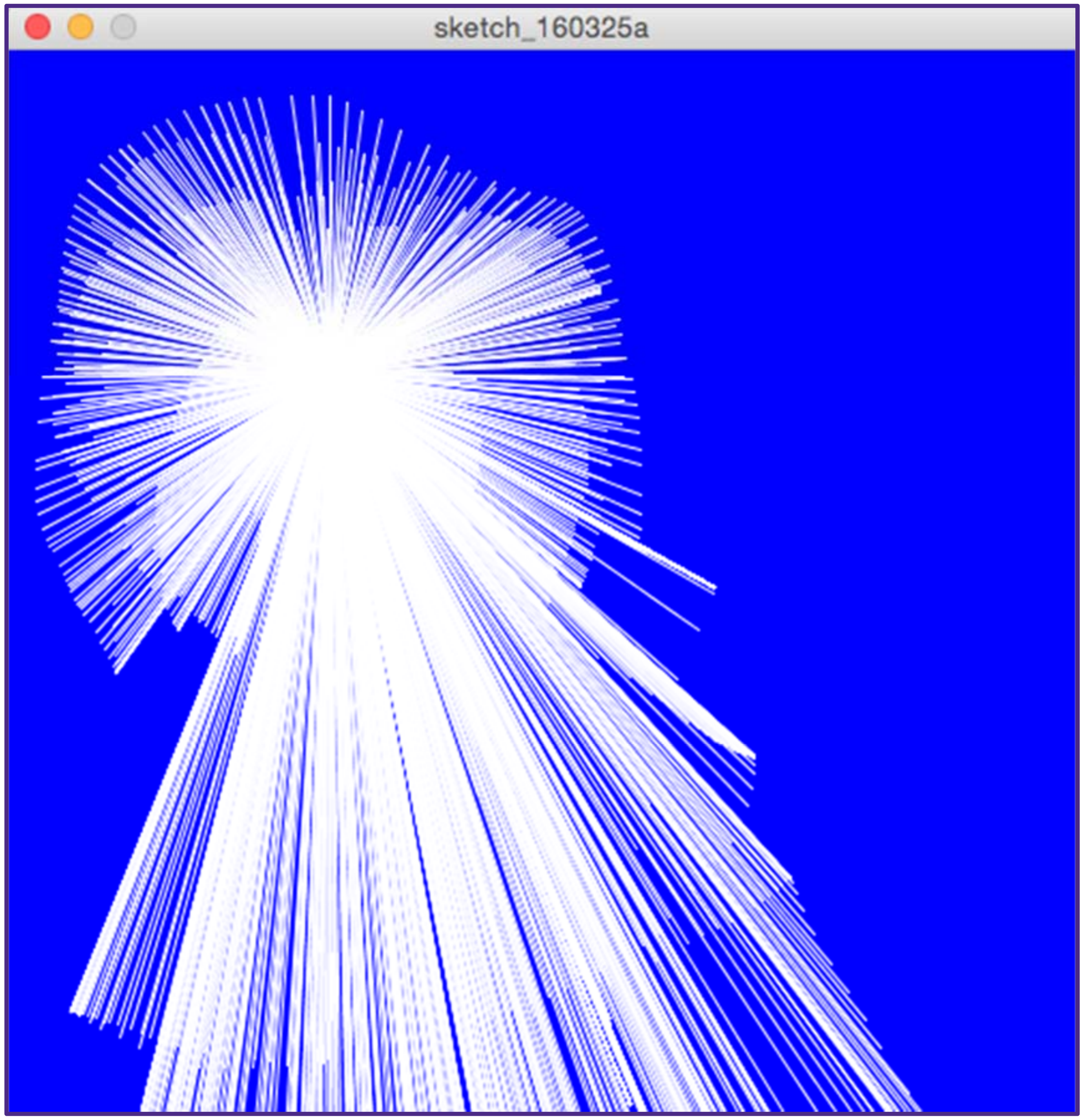
- It is both a programming *environment* (where you type) and a programming *language*
  - You are writing Java code, but they have made a lot of things easier

What You See

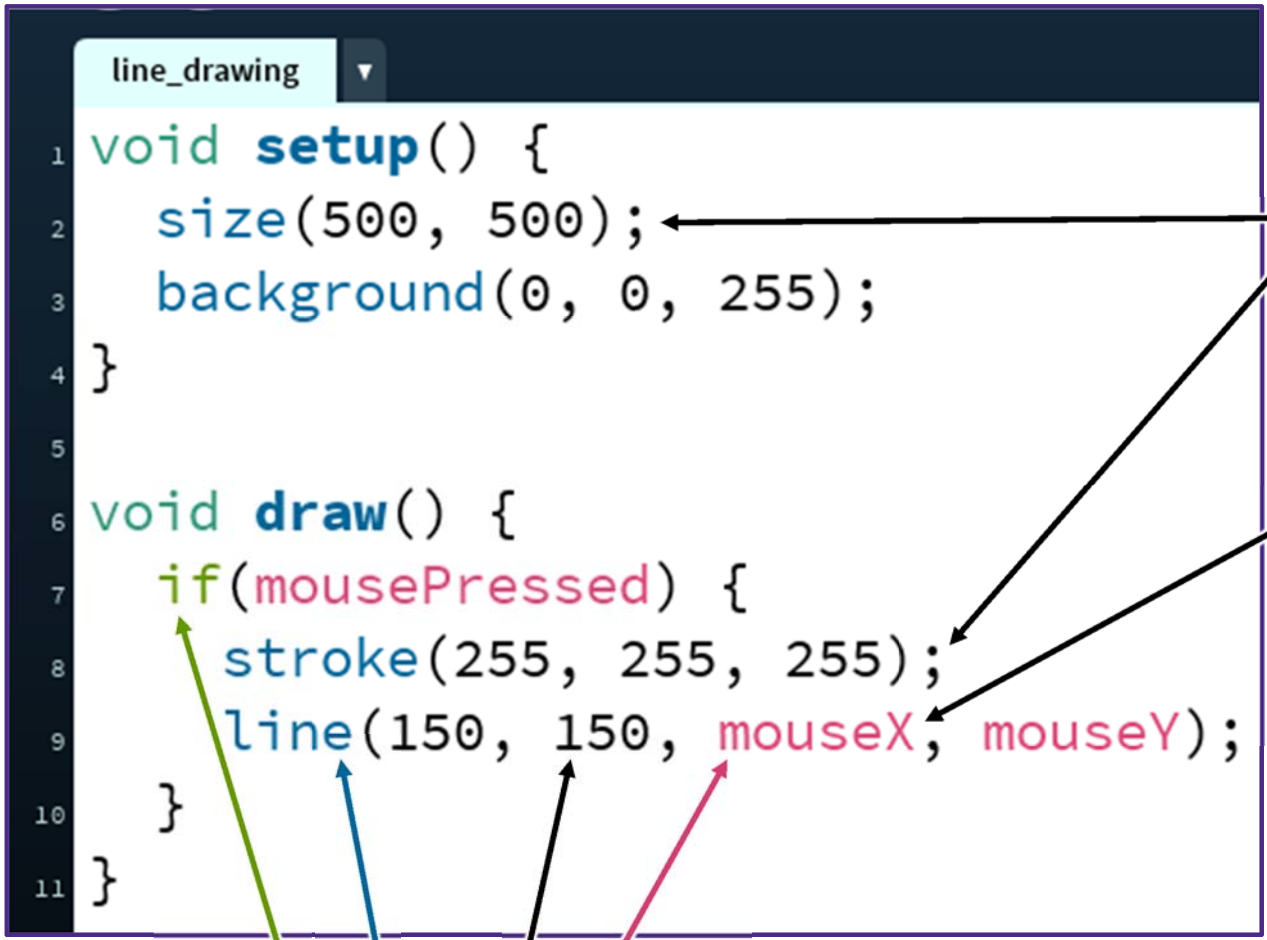

sketch_160325a | Processing 3.0.2
Java
sketch_160325a
Console
Errors


sketch_160325a

Interactive Line Drawing



Line Drawing Code



```
void setup() {
  size(500, 500);
  background(0, 0, 255);
}

void draw() {
  if(mousePressed) {
    stroke(255, 255, 255);
    line(150, 150, mouseX, mouseY);
  }
}
```

semi-colon indicates end of statement

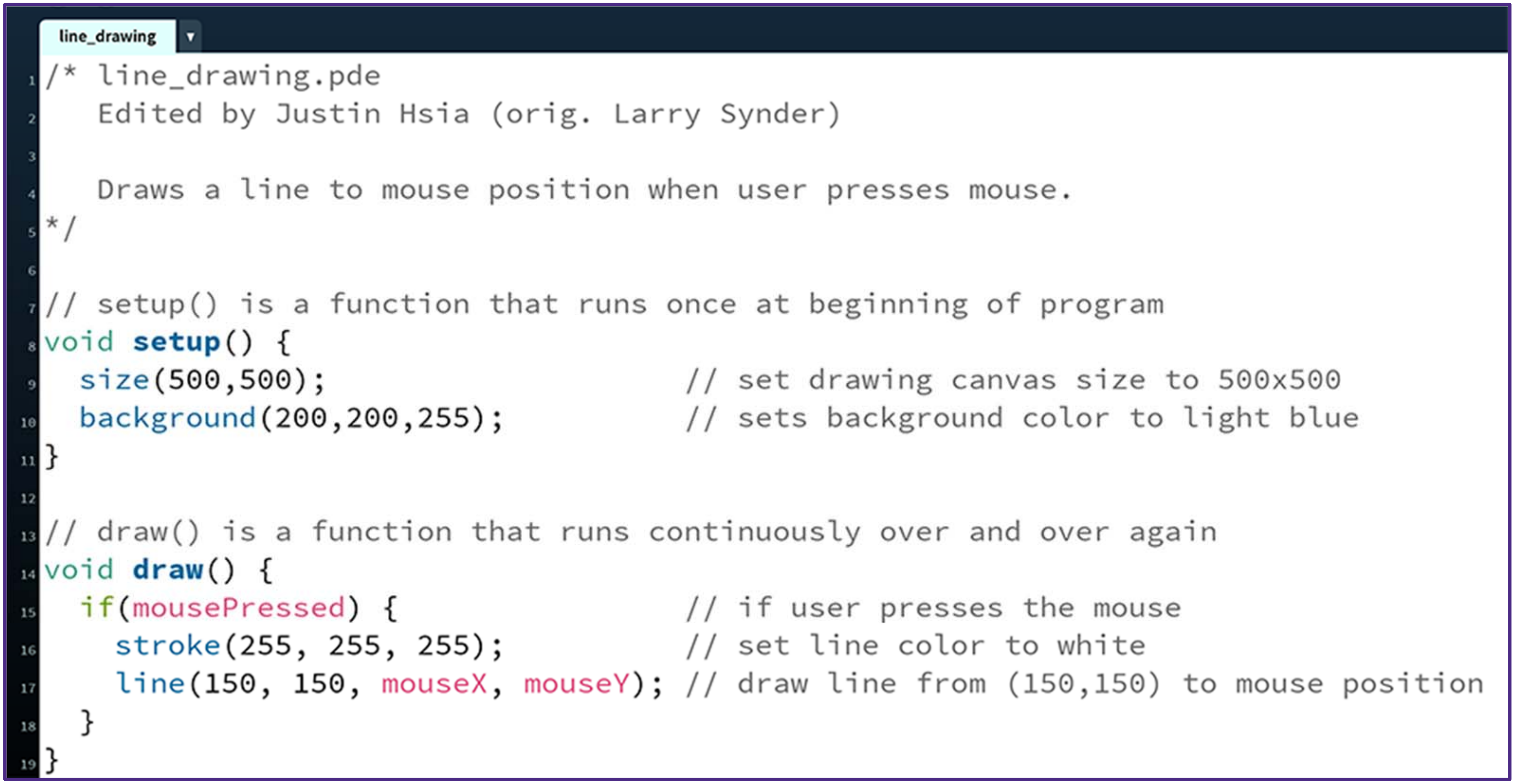
case-sensitive mouseX ≠ mousex

There is color coding

Other helpful *environment* features:

- Parentheses matching
- Error messages

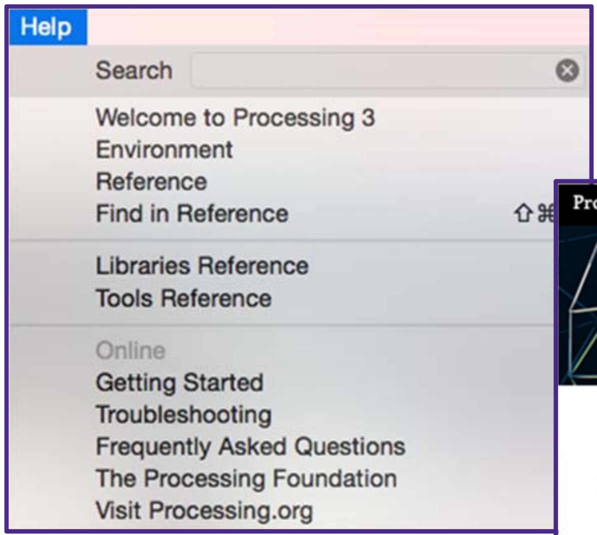
Comments Are Critical!!!



line_drawing

```
/* line_drawing.pde
   Edited by Justin Hsia (orig. Larry Synder)

   Draws a line to mouse position when user presses mouse.
*/

// setup() is a function that runs once at beginning of program
void setup() {
  size(500,500);                    // set drawing canvas size to 500x500
  background(200,200,255);          // sets background color to light blue
}

// draw() is a function that runs continuously over and over again
void draw() {
  if(mousePressed) {                // if user presses the mouse
    stroke(255, 255, 255);          // set line color to white
    line(150, 150, mouseX, mouseY); // draw line from (150,150) to mouse position
  }
}
```

The Processing Reference



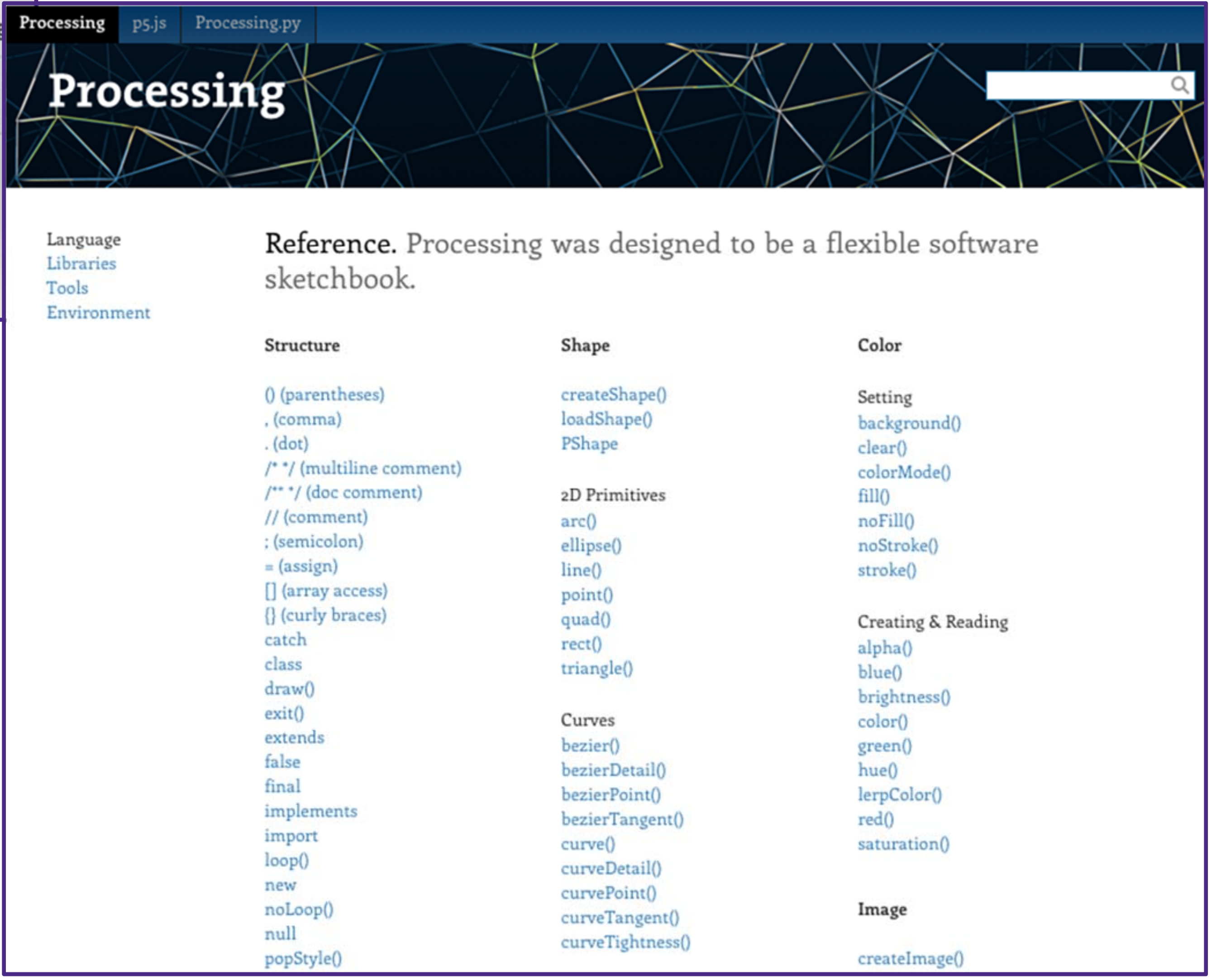
Help

Search

Welcome to Processing 3
Environment
Reference
Find in Reference ⇧⌘

Libraries Reference
Tools Reference

Online
Getting Started
Troubleshooting
Frequently Asked Questions
The Processing Foundation
Visit Processing.org



Processing p5.js Processing.py

Processing

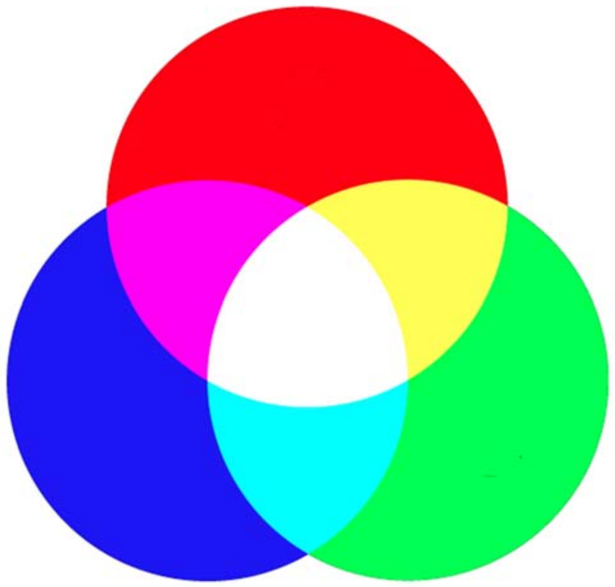
Language
Libraries
Tools
Environment

Reference. Processing was designed to be a flexible software sketchbook.

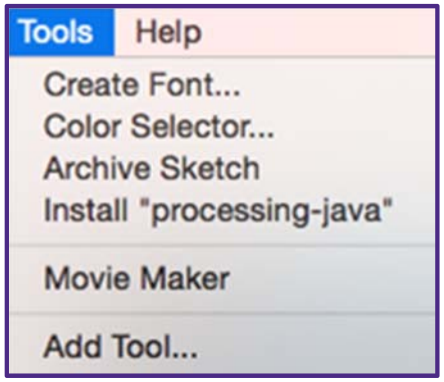
Structure	Shape	Color
() (parentheses)	createShape()	Setting
, (comma)	loadShape()	background()
. (dot)	PShape	clear()
/* */ (multiline comment)		colorMode()
/** */ (doc comment)	2D Primitives	fill()
// (comment)	arc()	noFill()
; (semicolon)	ellipse()	noStroke()
= (assign)	line()	stroke()
[] (array access)	point()	
{} (curly braces)	quad()	Creating & Reading
catch	rect()	alpha()
class	triangle()	blue()
draw()		brightness()
exit()	Curves	color()
extends	bezier()	green()
false	bezierDetail()	hue()
final	bezierPoint()	lerpColor()
implements	bezierTangent()	red()
import	curve()	saturation()
loop()	curveDetail()	
new	curvePoint()	Image
noLoop()	curveTangent()	
null	curveTightness()	createImage()
popStyle()		

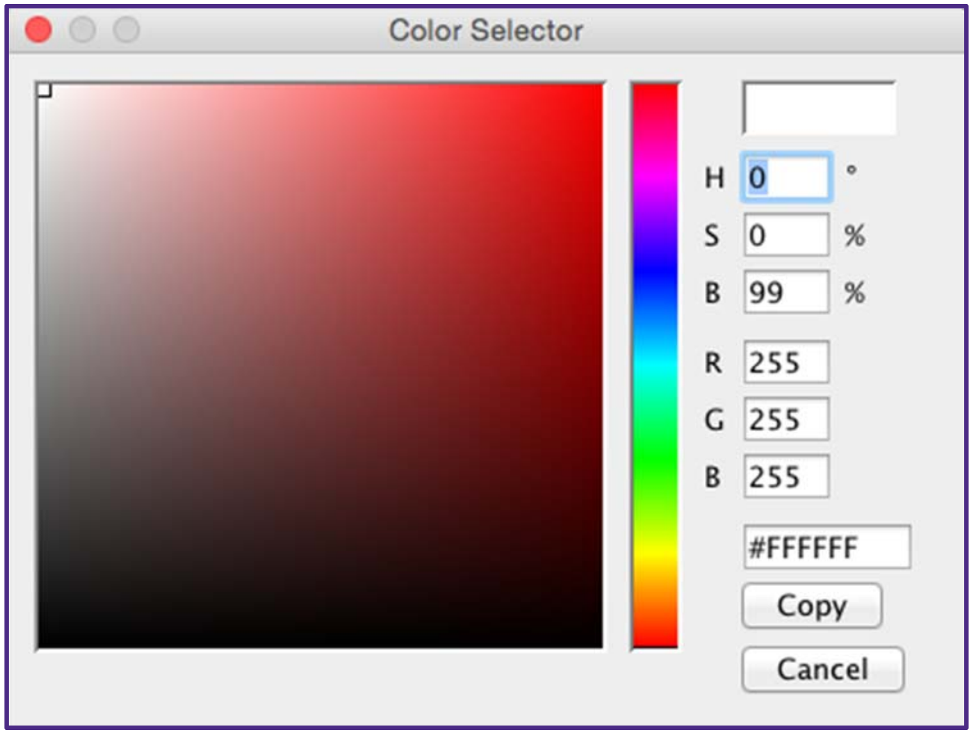
Understanding Color

- In electronic systems, color specified using the RGB color model
  - **R**ed, **G**reen, **B**lue
- Each pixel on your screen is made up of 3 tiny lights, one red, one green, one blue
- Specify the intensity of each light using an integer between 0 and 255
  - 0 is completely off
  - 255 is highest intensity



Processing's Color Selector





H 0 °
S 0 %
B 99 %
R 255
G 255
B 255
#FFFFFF

Guess the Color

```
color(  R,   G,   B);
color(255,   0,   0);
color(  0, 255,   0);
color(  0,   0, 255);
color(  0,   0,   0);
color(255, 255, 255);
color(255, 255,   0);
color(255,   0, 255);
color(  0, 255, 255);
```

Guess the Color

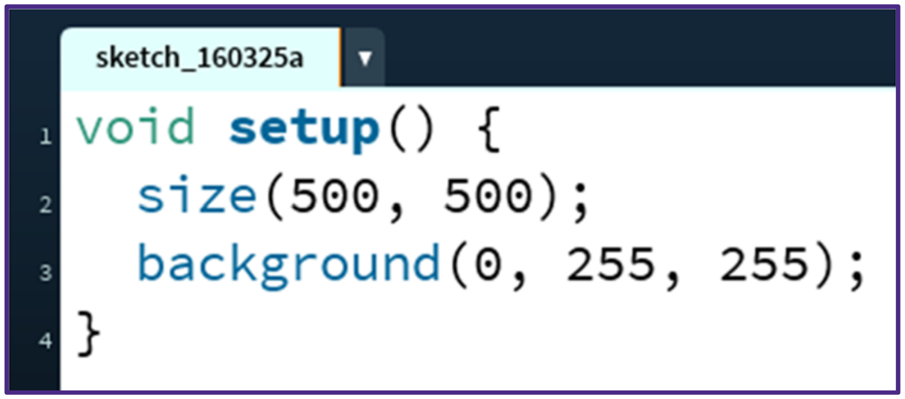
```
❖ color(  R,   G,   B);
❖ color(255,   0,   0); // R fully on
❖ color(  0, 255,   0); // G fully on
❖ color(  0,   0, 255); // B fully on
❖ color(  0,   0,   0); // all off
❖ color(255, 255, 255); // all fully on
❖ color(255, 255,   0); // R,G fully on
❖ color(255,   0, 255); // R,B fully on
❖ color(  0, 255, 255); // G,B fully on
```

Guess the Color

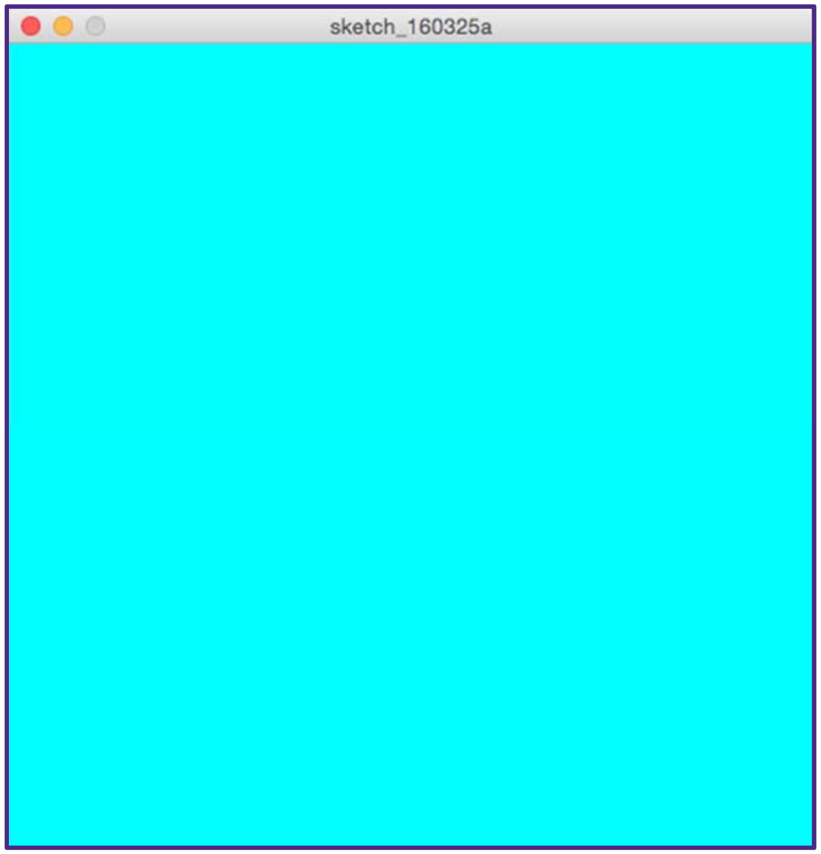
- `color(  R,   G,   B);`
- `color(255,   0,   0); // red`
- `color(  0, 255,   0); // green`
- `color(  0,   0, 255); // blue`
- `color(  0,   0,   0); // black`
- `color(255, 255, 255); // white`
- `color(255, 255,   0); // yellow`
- `color(255,   0, 255); // magenta`
- `color(  0, 255, 255); // cyan`

Color Functions

- `background(R, G, B);`
  - Sets the background color of the drawing canvas

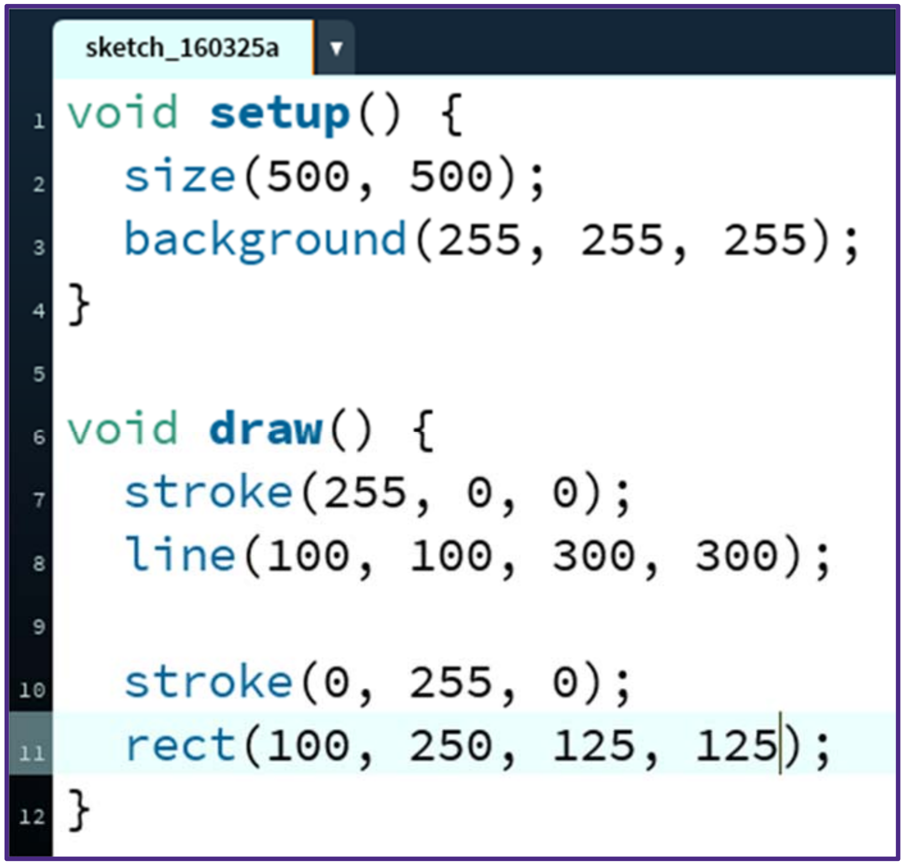


```
void setup() {
  size(500, 500);
  background(0, 255, 255);
}
```

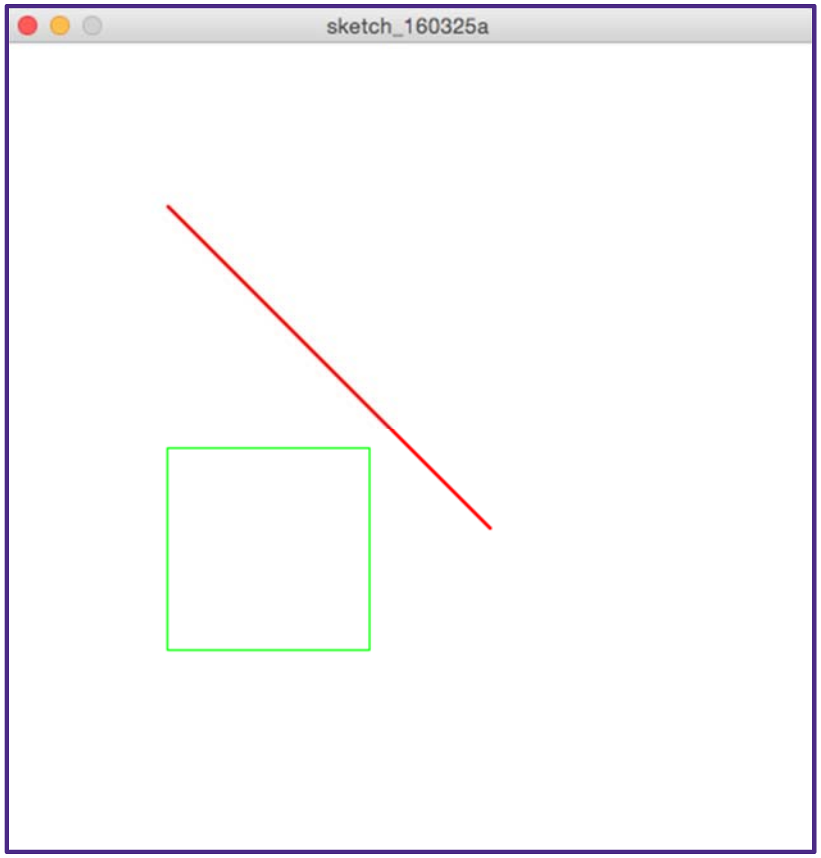


Color Functions

- `stroke(R, G, B);`
  - Sets the color of the stroke of a *line* or *line around a shape*
  - Can change line size using `strokeWeight(#);`

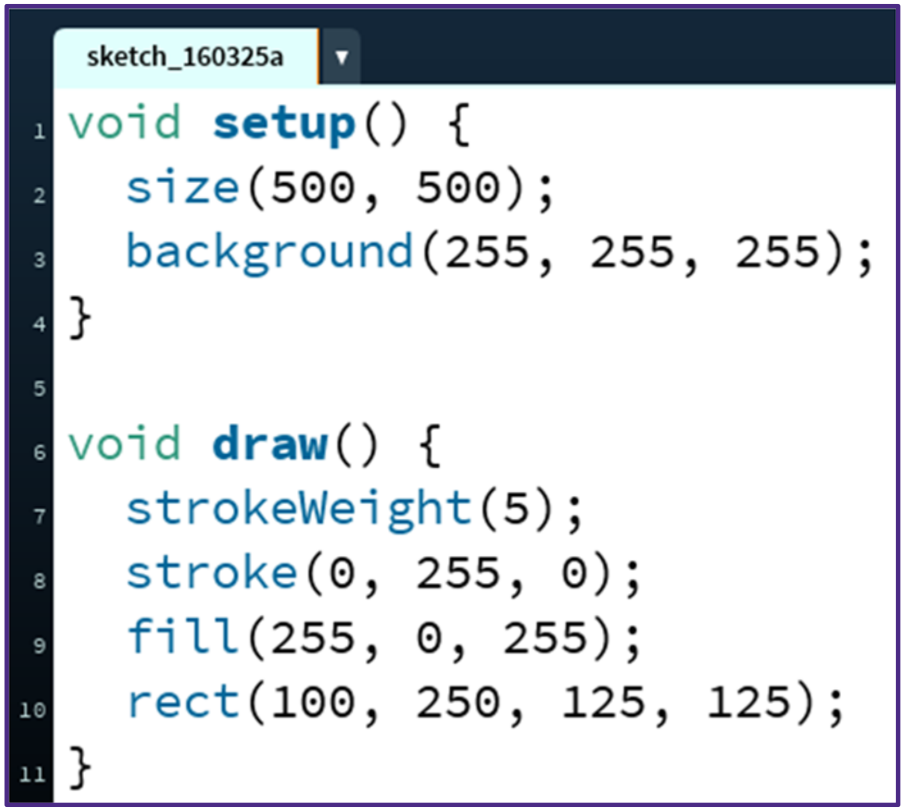


```
void setup() {
  size(500, 500);
  background(255, 255, 255);
}

void draw() {
  stroke(255, 0, 0);
  line(100, 100, 300, 300);

  stroke(0, 255, 0);
  rect(100, 250, 125, 125);
}
```

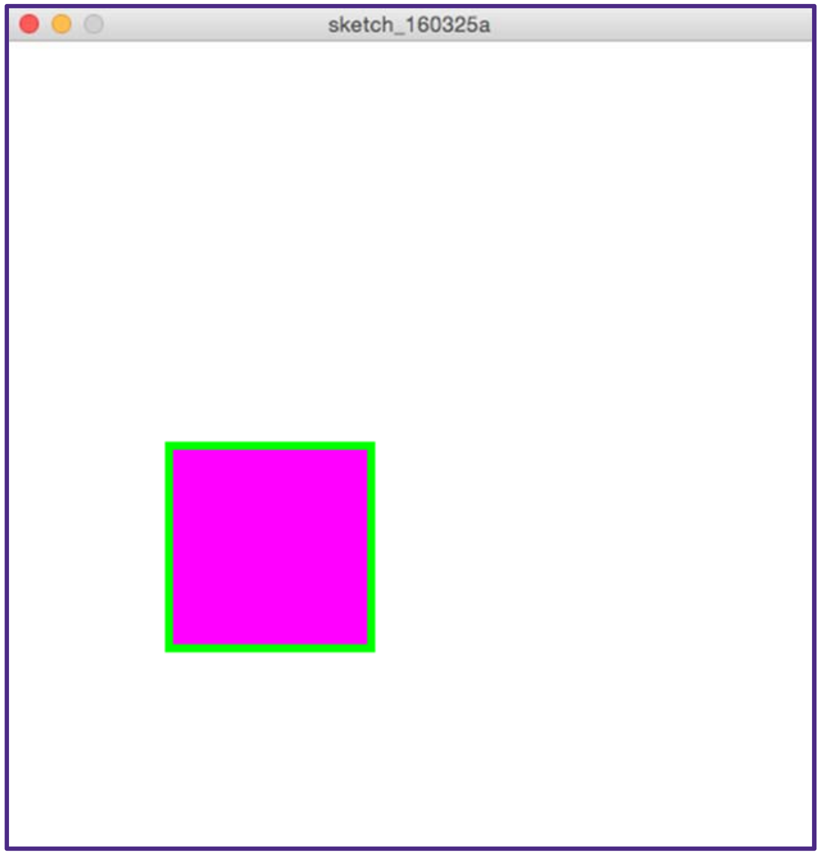


Color Functions

- `fill(R, G, B);`
  - Sets the *inside* color of a shape (**note:** you cannot fill a line)

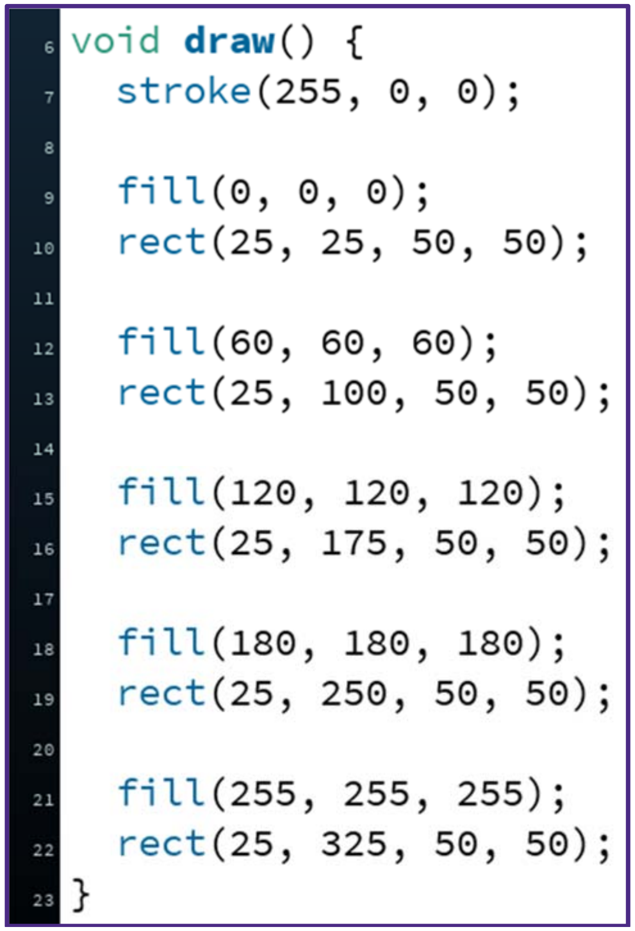


```
void setup() {
  size(500, 500);
  background(255, 255, 255);
}

void draw() {
  strokeWeight(5);
  stroke(0, 255, 0);
  fill(255, 0, 255);
  rect(100, 250, 125, 125);
}
```

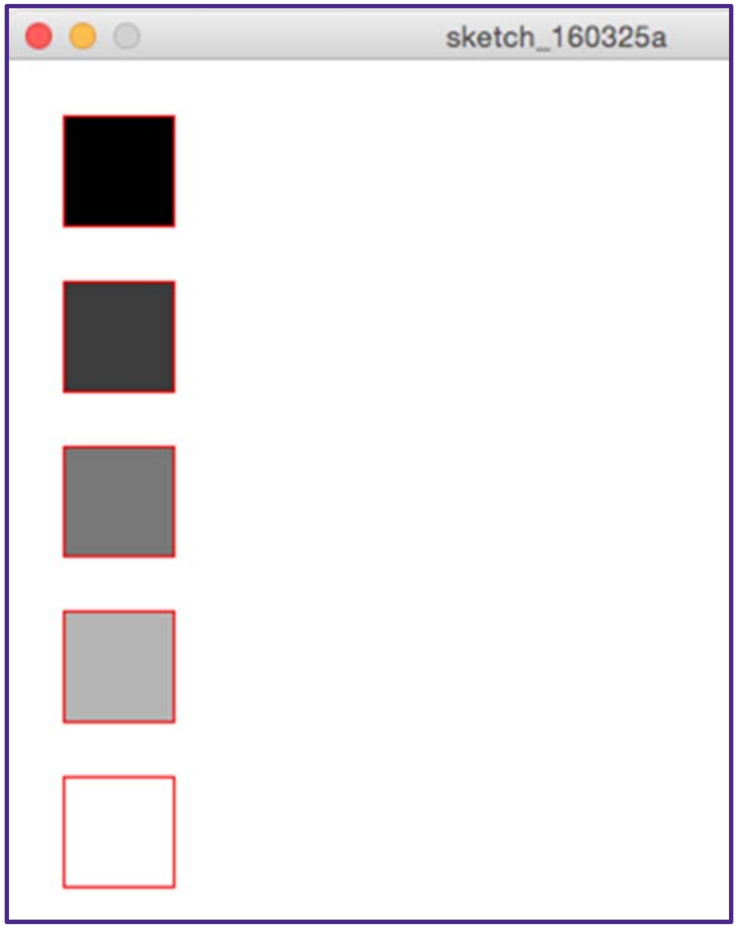


Color: “Grays"

- When the values for RGB are all the same, then the color will be white, black, or some shade of gray

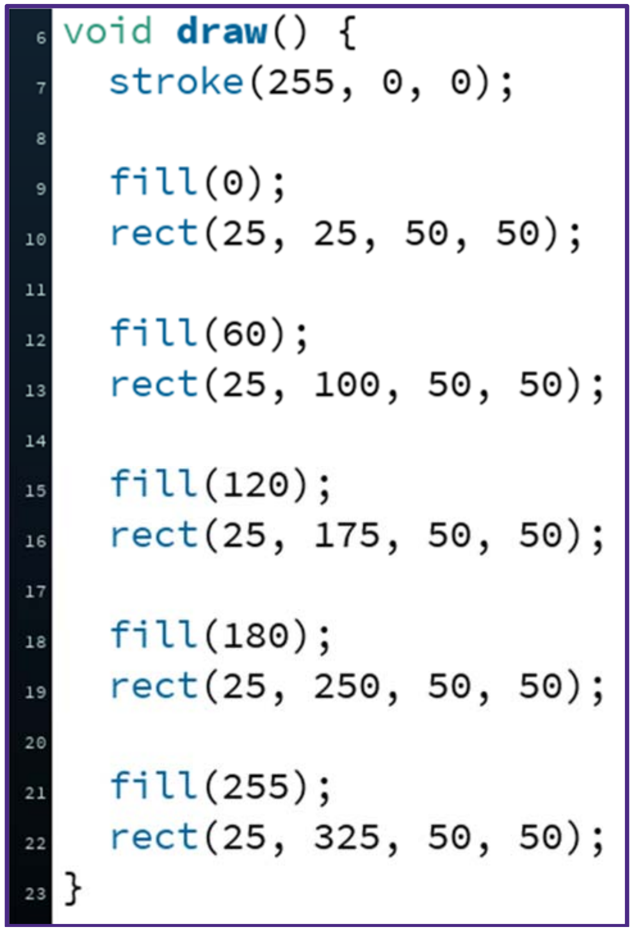


```
void draw() {
  stroke(255, 0, 0);

  fill(0, 0, 0);
  rect(25, 25, 50, 50);

  fill(60, 60, 60);
  rect(25, 100, 50, 50);

  fill(120, 120, 120);
  rect(25, 175, 50, 50);

  fill(180, 180, 180);
  rect(25, 250, 50, 50);

  fill(255, 255, 255);
  rect(25, 325, 50, 50);
}
```

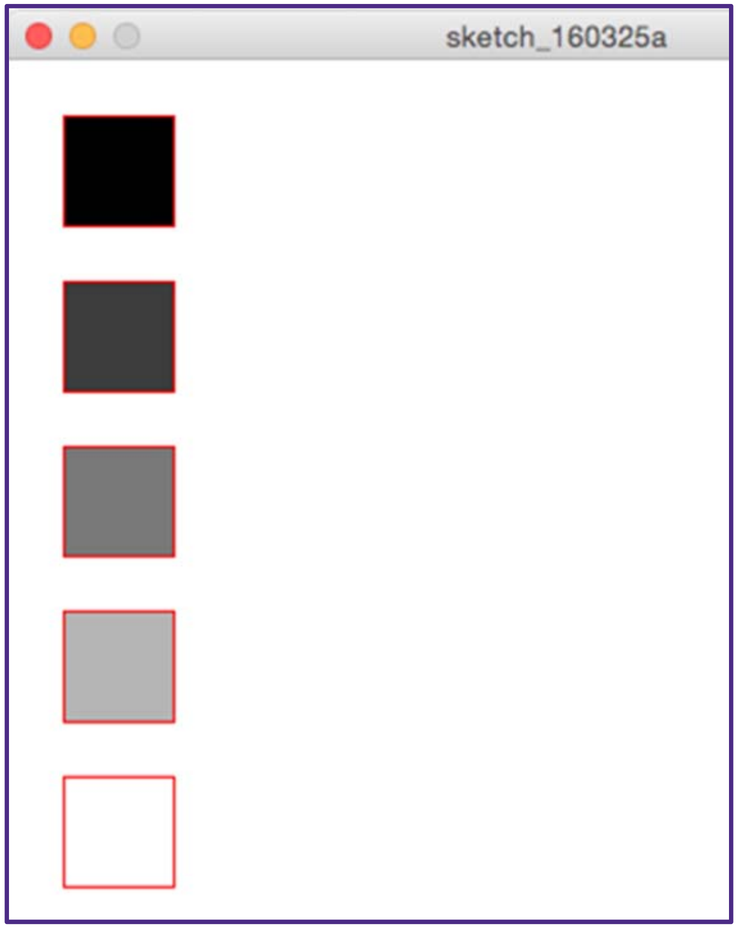


Color: “Grays"

- When the values for RGB are all the same, then the color will be white, black, or some shade of gray
  - For brevity, can specify just a single number instead



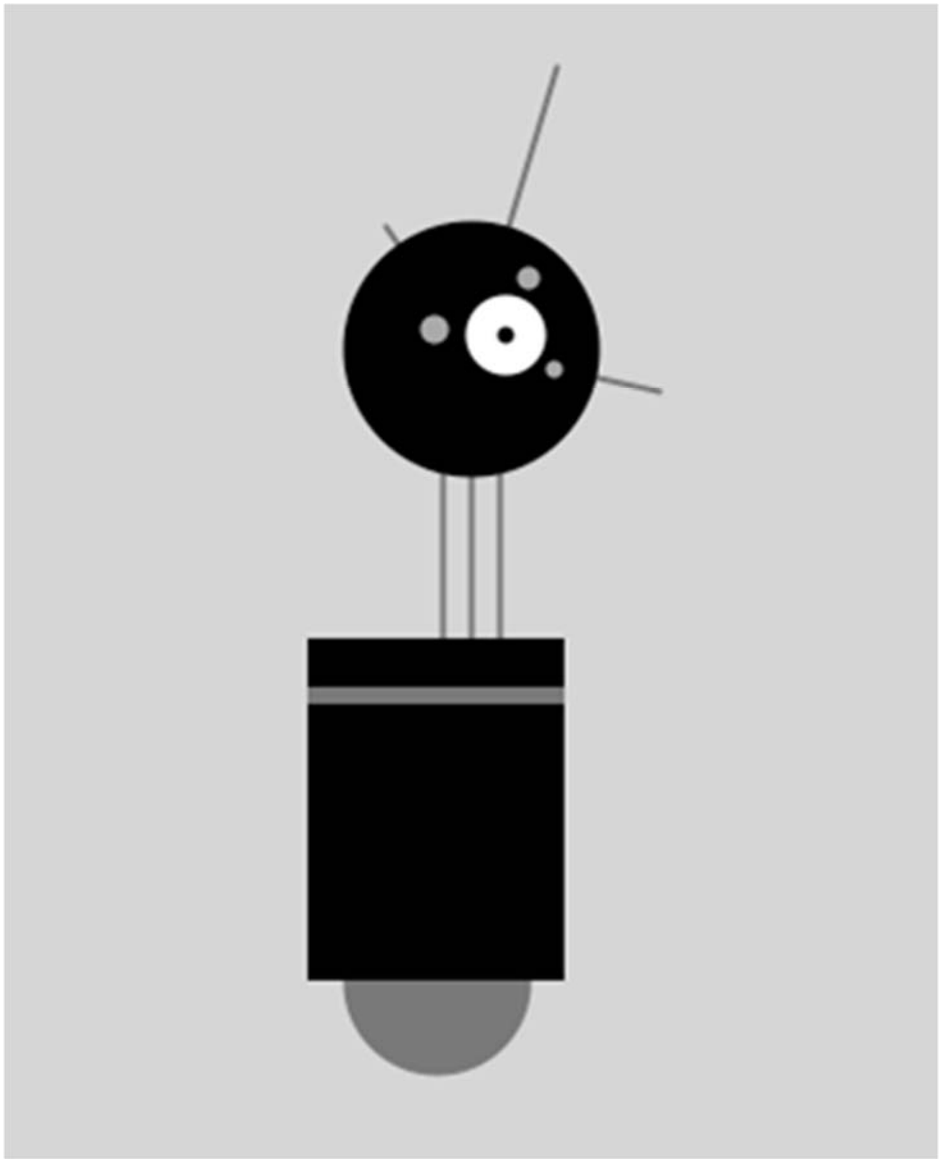
```
void draw() {
  stroke(255, 0, 0);

  fill(0);
  rect(25, 25, 50, 50);

  fill(60);
  rect(25, 100, 50, 50);

  fill(120);
  rect(25, 175, 50, 50);

  fill(180);
  rect(25, 250, 50, 50);

  fill(255);
  rect(25, 325, 50, 50);
}
```



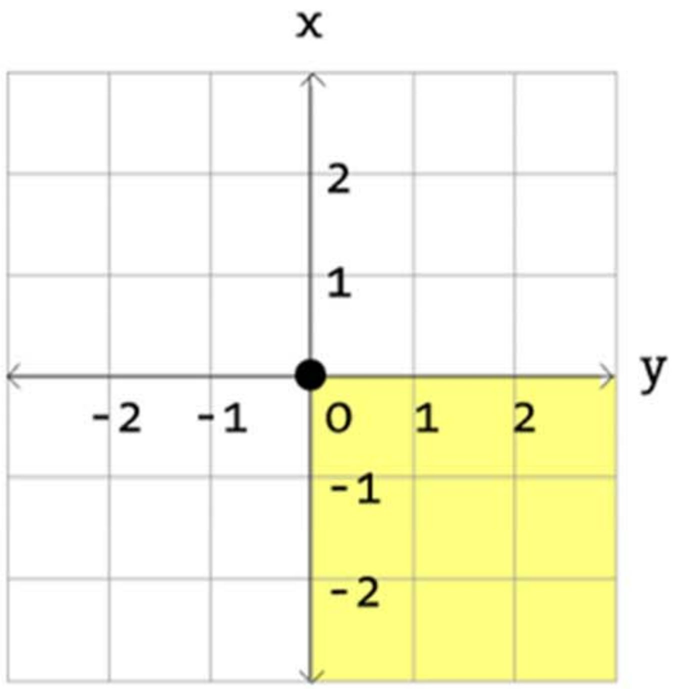
The Color "State" of Your Program

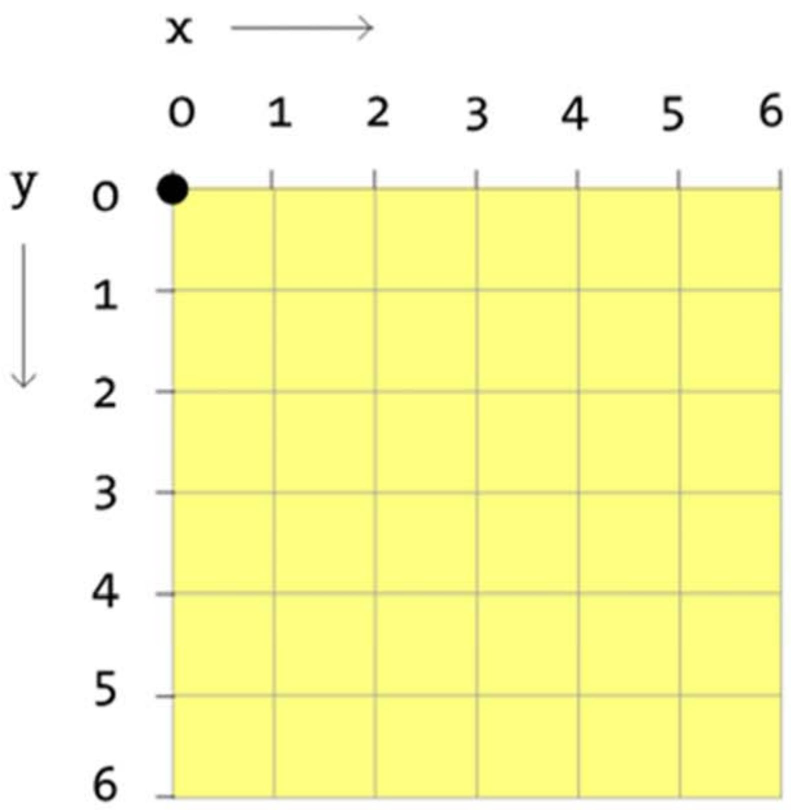
- Recall that programs are executed sequentially (*i.e.* instruction-by-instruction)
- `background()`, `stroke()`, and `fill()` apply to *all* subsequent drawing statements
  - Until a later call overrides
- Hidden color "state" that knows the current values of `background()`, `stroke()`, and `fill()`
  - In complex programs, can be difficult to keep track of
  - Early rule of thumb: always explicitly set colors before each drawing element

Assignment: Coloring a Robot

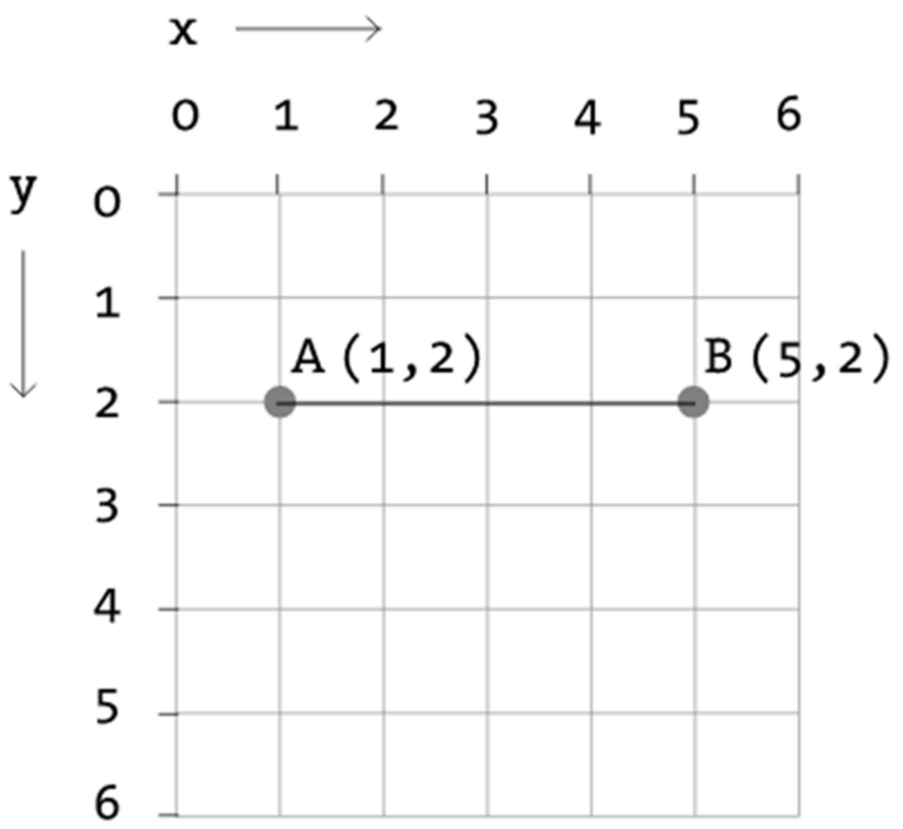


Coordinate System





Drawing: Line

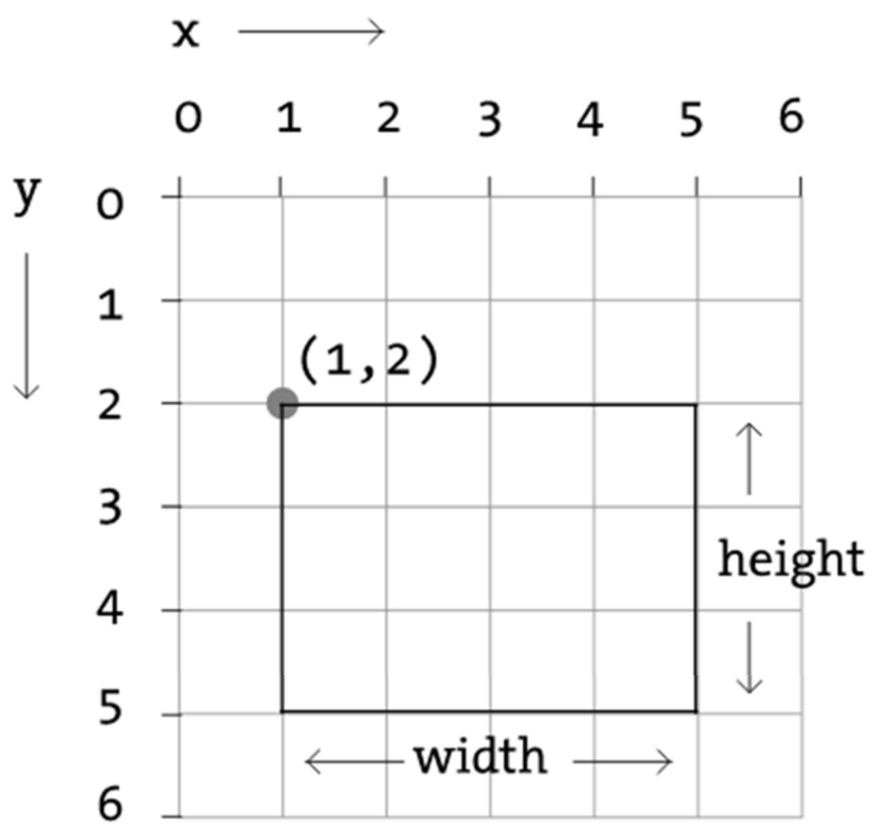


$\text{line}(\underbrace{x1, y1}_{\text{Point A}}, \underbrace{x2, y2}_{\text{Point B}});$

Example: line(1,2,5,2) ;

Drawing: Rectangle

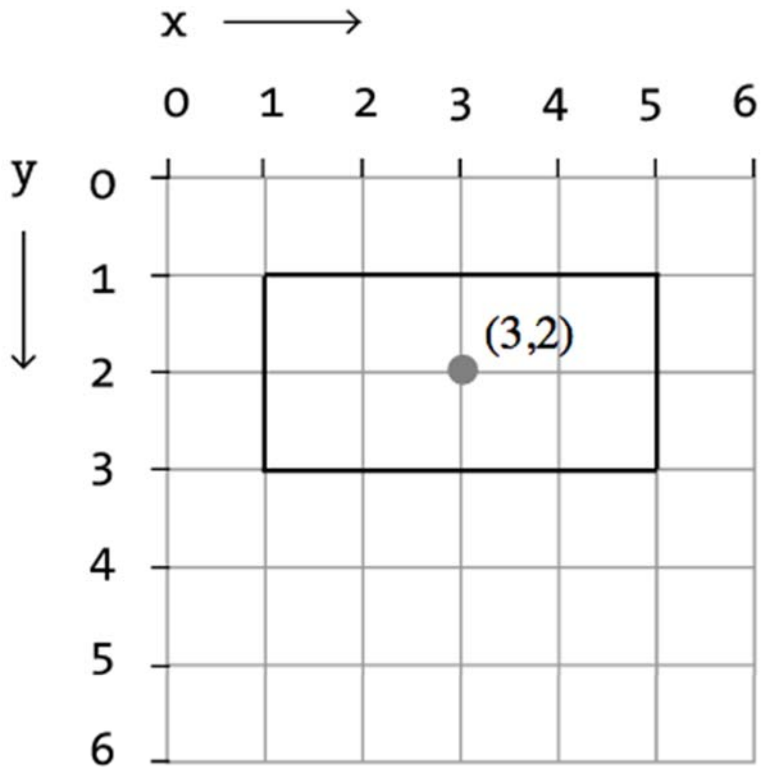
- Default *mode* is CORNER



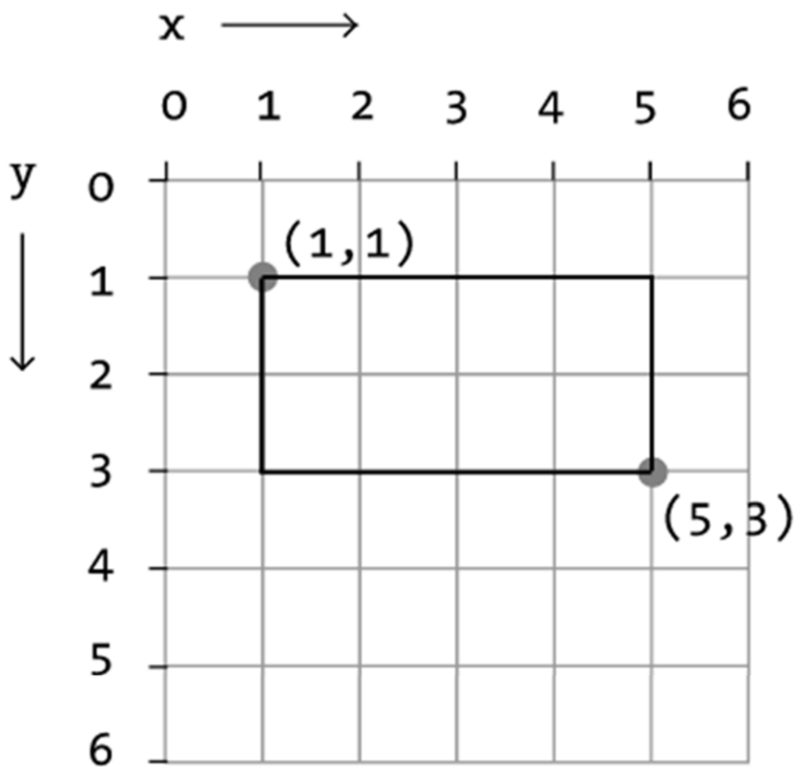
Example: `rect(1,2,4,3);`

Drawing: Additional Rect Modes

- CENTER
- CORNERS



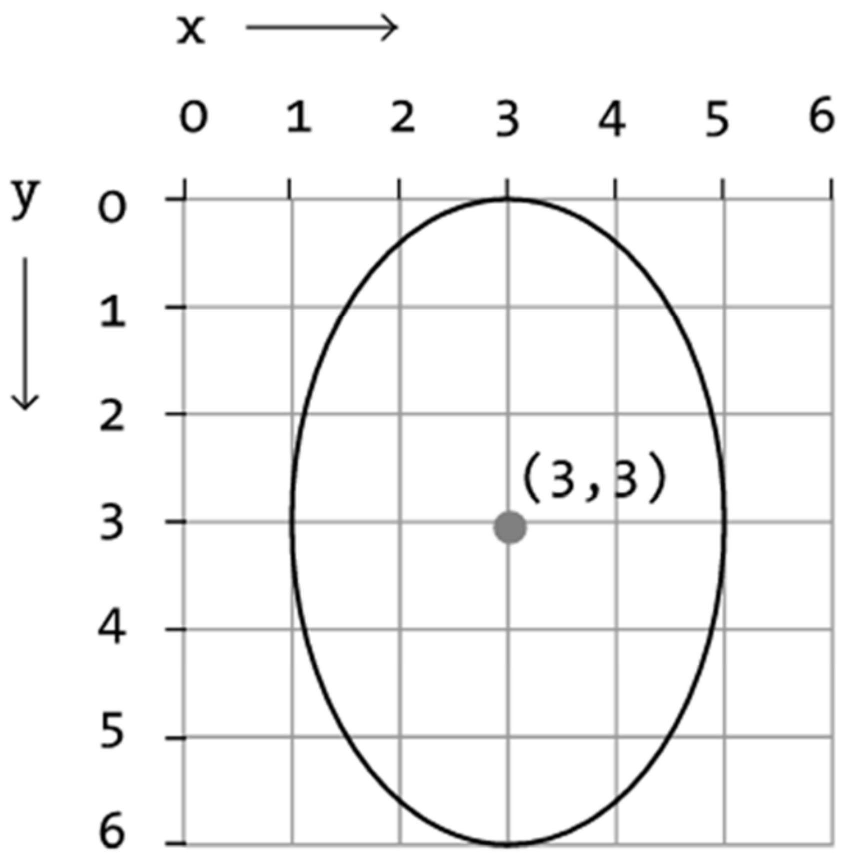
Example: rectMode(CENTER);
rect(3,2,4,2);



Example: rectMode(CORNERS);
rect(1,1,5,3) ;

Drawing: Ellipse/Circle

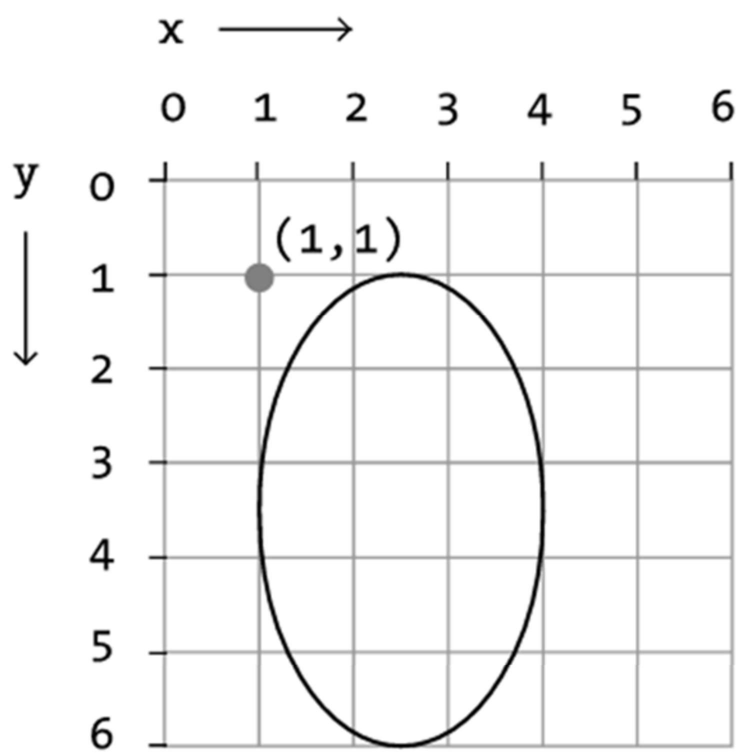
- Default *mode* is CENTER



Example: `ellipse(3,3,4,6);`

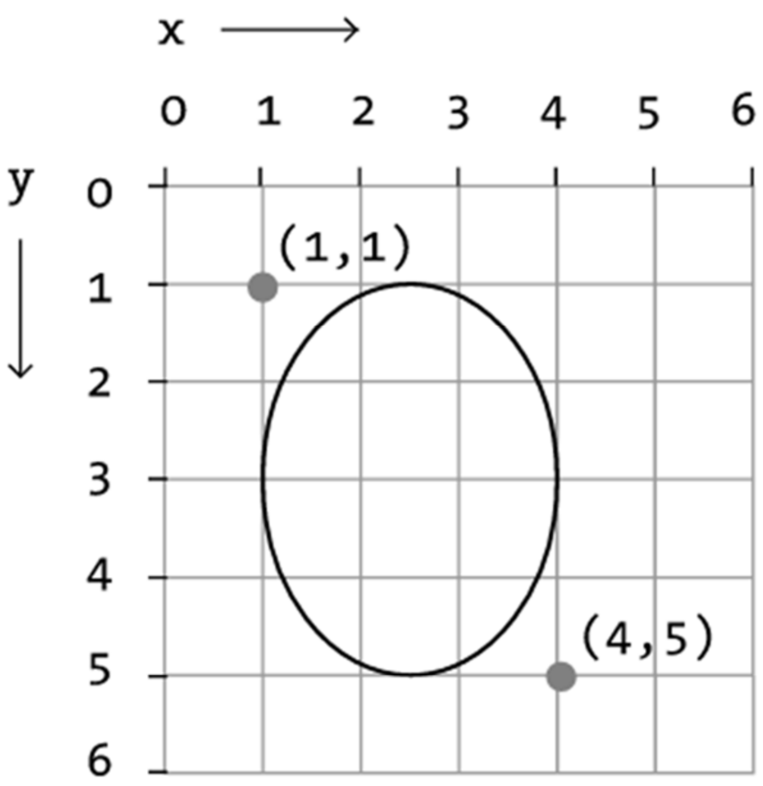
Drawing: Additional Ellipse Modes

- CORNER



Example: ellipseMode(CORNER);
ellipse(1,1,3,5);

- CORNERS

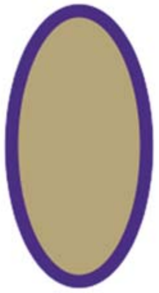


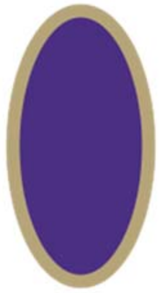
Example: ellipseMode(CORNERS);
ellipse(1,1,4,5) ;

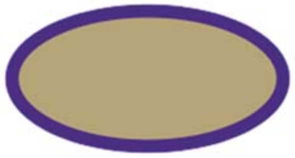
Peer Instruction Question

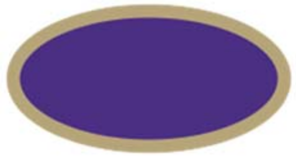
- Which of the following drawings corresponds to the Processing code below?
  - Vote at http://PollEv.com/justinh

```
strokeWeight(10);
stroke(75, 47, 131);           // UW purple
fill(183, 165, 122);           // UW gold
ellipse(100, 100, 100, 200); // CENTER mode
```

A. 

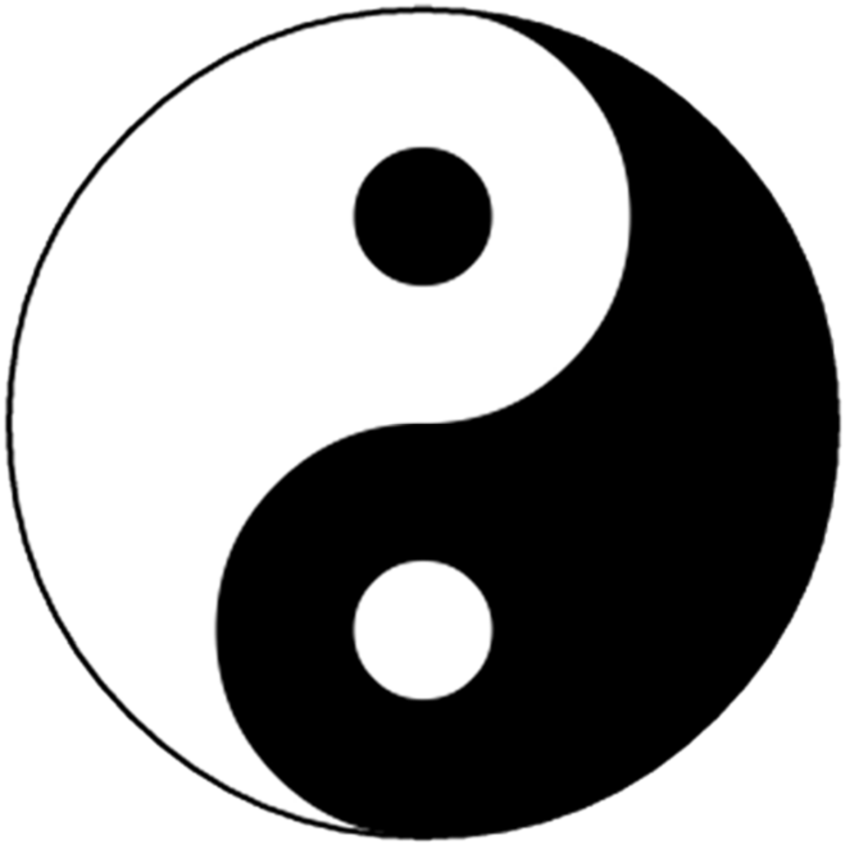
B. 

C. 

D. 

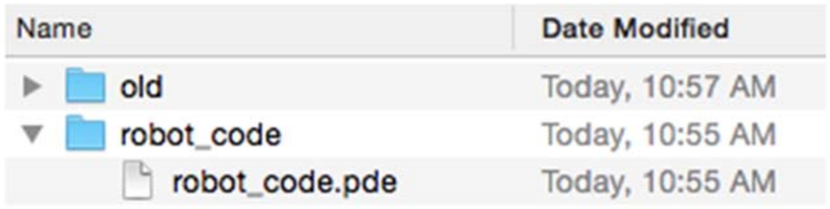
Lab: Taijitu

- How do you build a complex drawing out of these simple shapes?

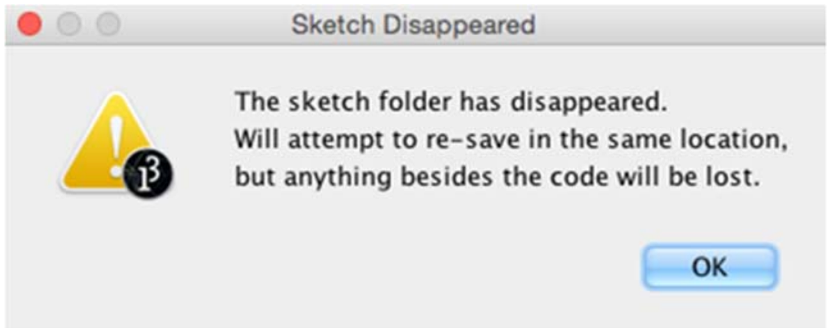


Aside: Processing Files

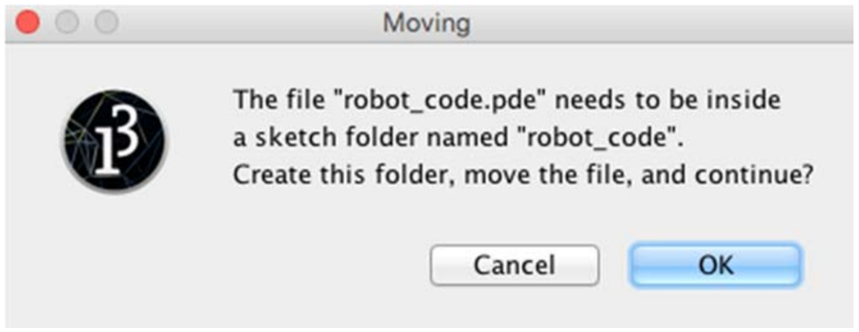
- Processing files have extension `.pde`
  - File names *cannot* contain dashes (`-`)
- To run a Processing file, it *must* be in a folder of the same name
  - If it's not, then Processing will create the folder for you



Name	Date Modified
▶ old	Today, 10:57 AM
▼ robot_code	Today, 10:55 AM
robot_code.pde	Today, 10:55 AM



The sketch folder has disappeared.
Will attempt to re-save in the same location,
but anything besides the code will be lost.



The file "robot_code.pde" needs to be inside
a sketch folder named "robot_code".
Create this folder, move the file, and continue?