

Processing and Drawing

CSE 120 Winter 2019

Instructor:

Justin Hsia

Teaching Assistants:

Ann Shan,

Sam Wolfson,

Eunia Lee,

Travis McGaha

Pei Lee Yap,

There are now 2,823 emoji. Meet the former Apple intern who helped design the original 500

“The latest Apple software update, iOS12.1, added 157 new emoji to the ever-growing library. This brings the total number to 2,823, which includes all the variations for elements like skin tone and gender. It may be hard to imagine a world before these little icons, but it was only a decade ago that they got their start.

“Meet Angela Guzman, a former Apple design intern who worked with her mentor Raymond Sepulveda to create about 500 of the original emoji, ... including the happy poop, the colorful hearts and a seemingly endless variety of facial expressions.”

- <https://www.cnbc.com/2018/12/27/this-former-apple-intern-helped-design-the-original-500-emoji.html>



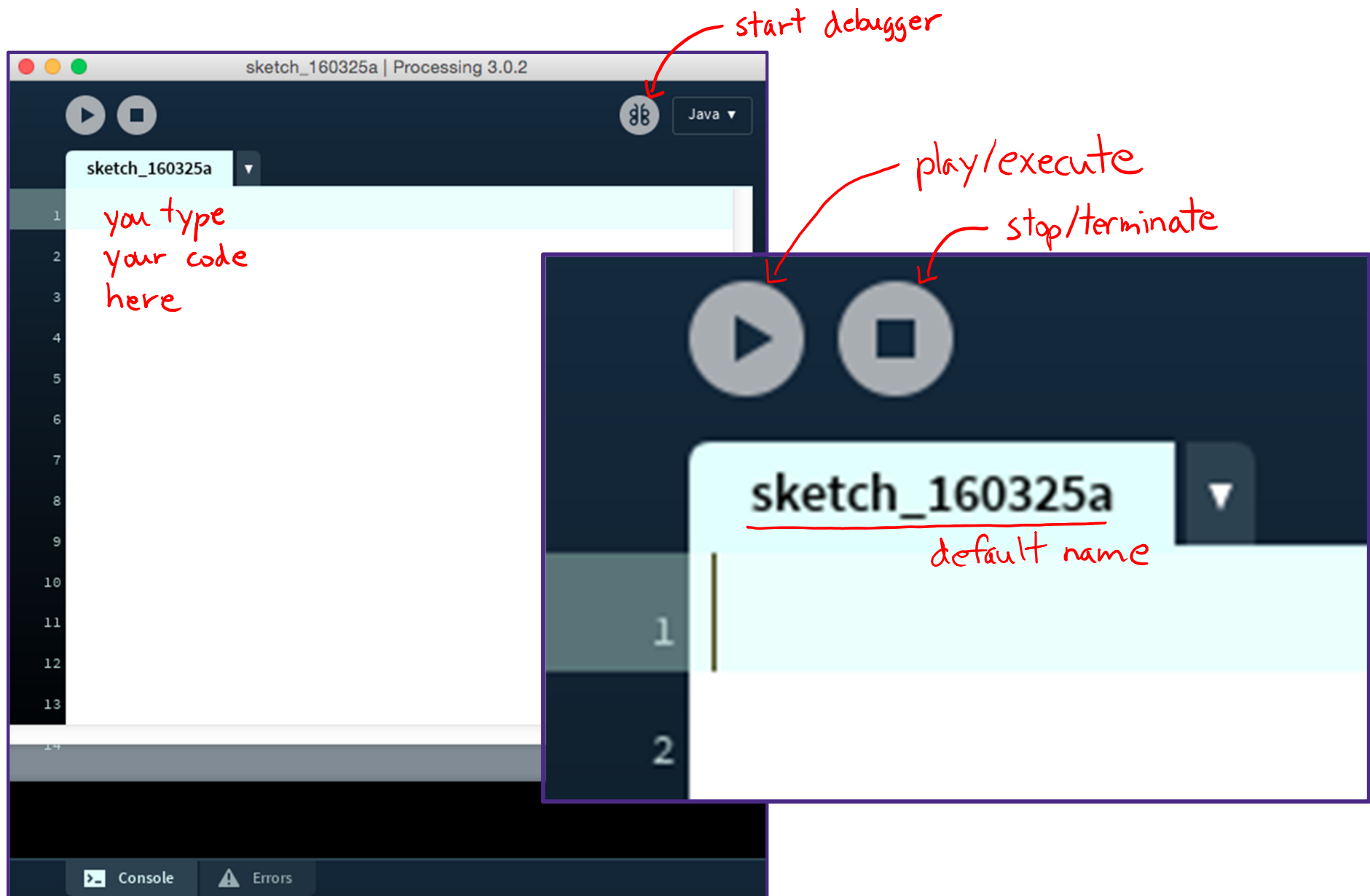
Administrivia

- ❖ Assignments:
 - Lightbot Functions [hw] due today *before 11:59 pm* (1/14)
 - Taijitu [lab] due by end of Thursday (1/17)
- ❖ “Big Ideas” lecture this week: Algorithms
 - Reading due before lab on Thursday (1/17)
- ❖ **Register on Piazza** (7 of you still haven't)
- ❖ Grading and Grades
 - Reading Check 1 and Personal Values scores released
 - Assignment have rubrics on Canvas
 - Final grades will be curved, but not to a strict curve

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

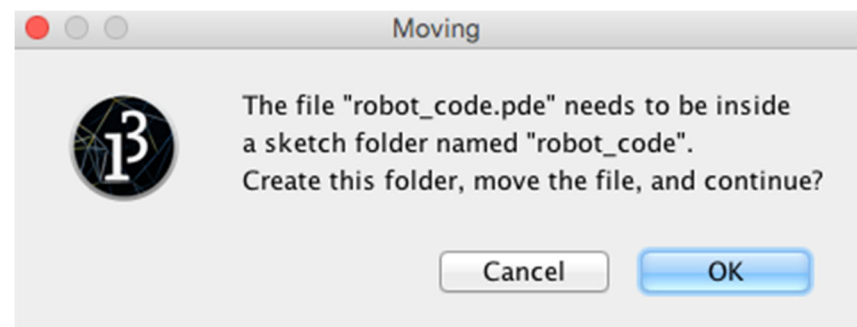
The Processing Coding Environment



Aside: Processing Files

- ❖ Processing files have extension `.pde`
 - File names *cannot* contain dashes (-) *use underscore (_) instead*
- ❖ 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
▶ folder old	Today, 10:57 AM
▼ folder robot_code	Today, 10:55 AM
file robot_code.pde	Today, 10:55 AM



Text-Based Programming Basics

```
line_drawing
1 void setup() {
2   size(500, 500);
3   background(0, 0, 255);
4 }
5
6 void draw() {
7   if(mousePressed) {
8     stroke(255, 255, 255);
9     line(150, 150, mouseX, mouseY);
10  }
11 }
```

semi-colon indicates end of statement

case-sensitive
mouseX ≠ mousex

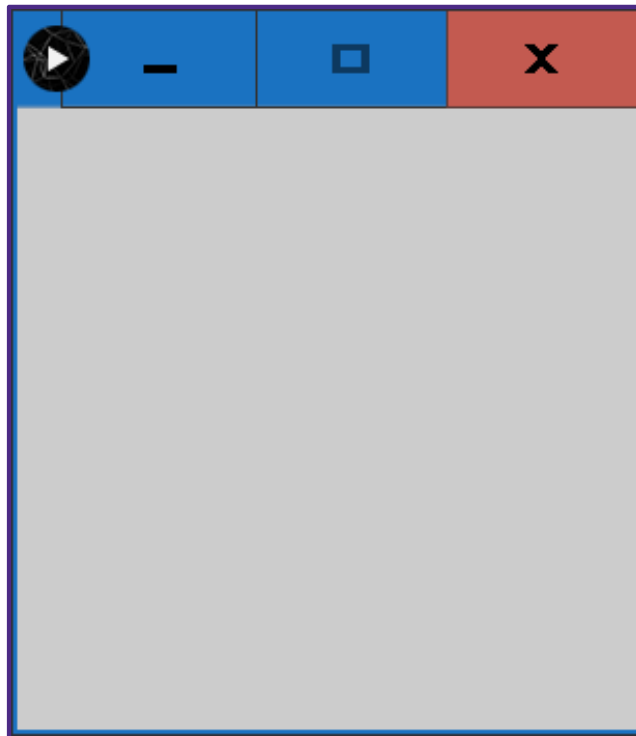
There is color coding

Other helpful *environment* features:

- Parentheses matching
- Error messages

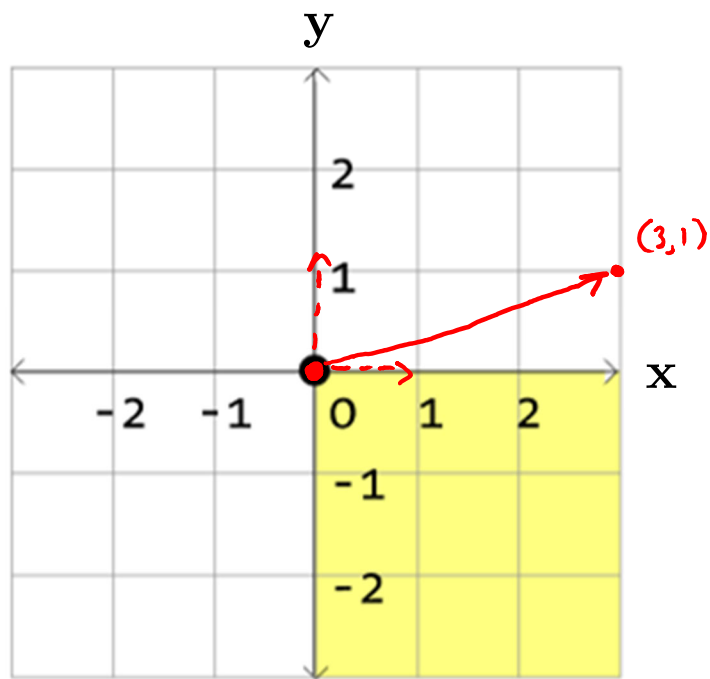
The Drawing Canvas

- ❖ Defines the space on which you can draw
 - `size` (`width`, `height`);
 - Anything drawn *off* of the canvas won't be visible



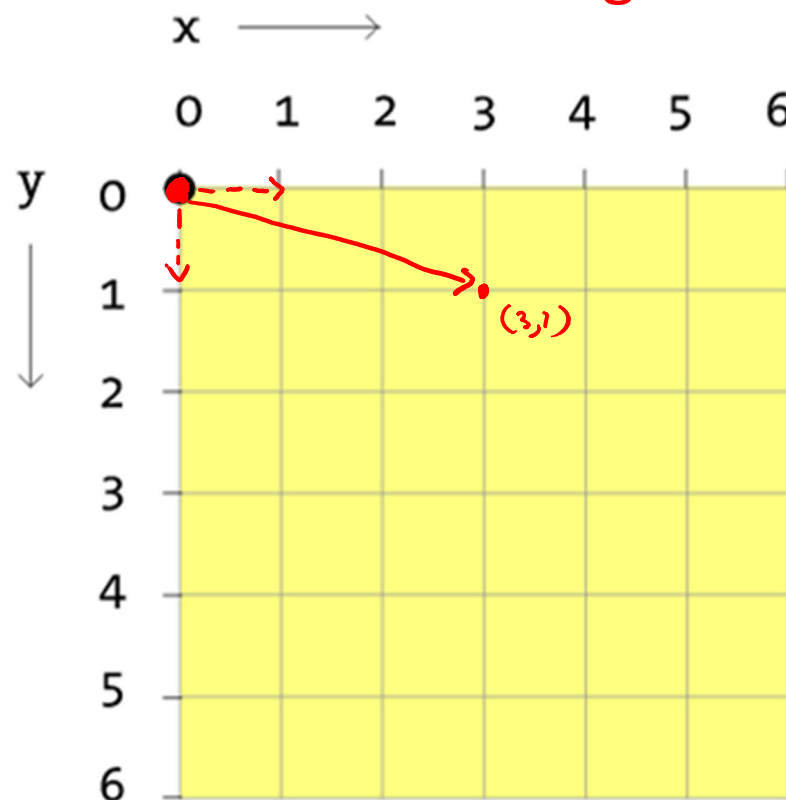
Coordinate System

Math



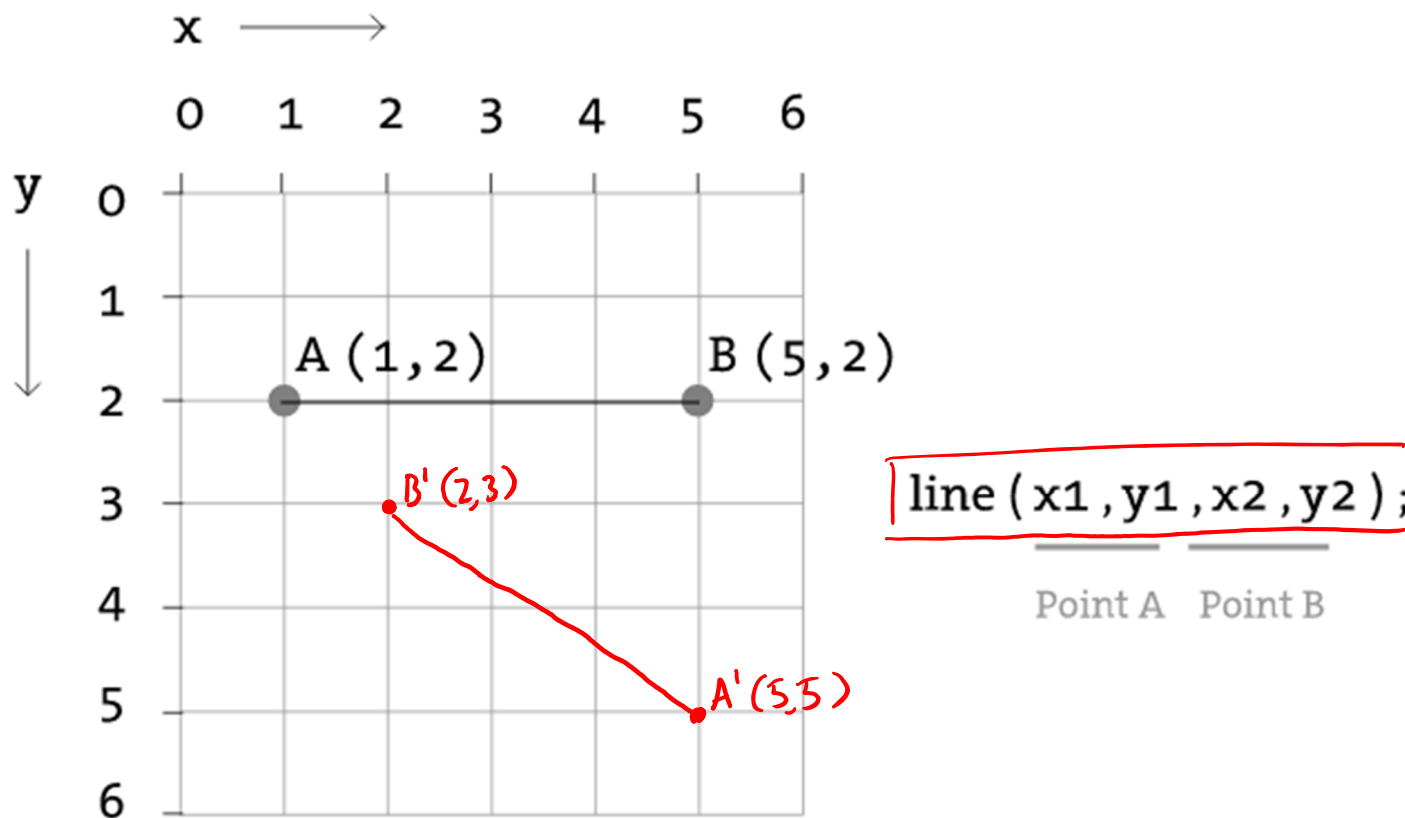
origin (0,0) is center

Processing



origin (0,0) is upper-left

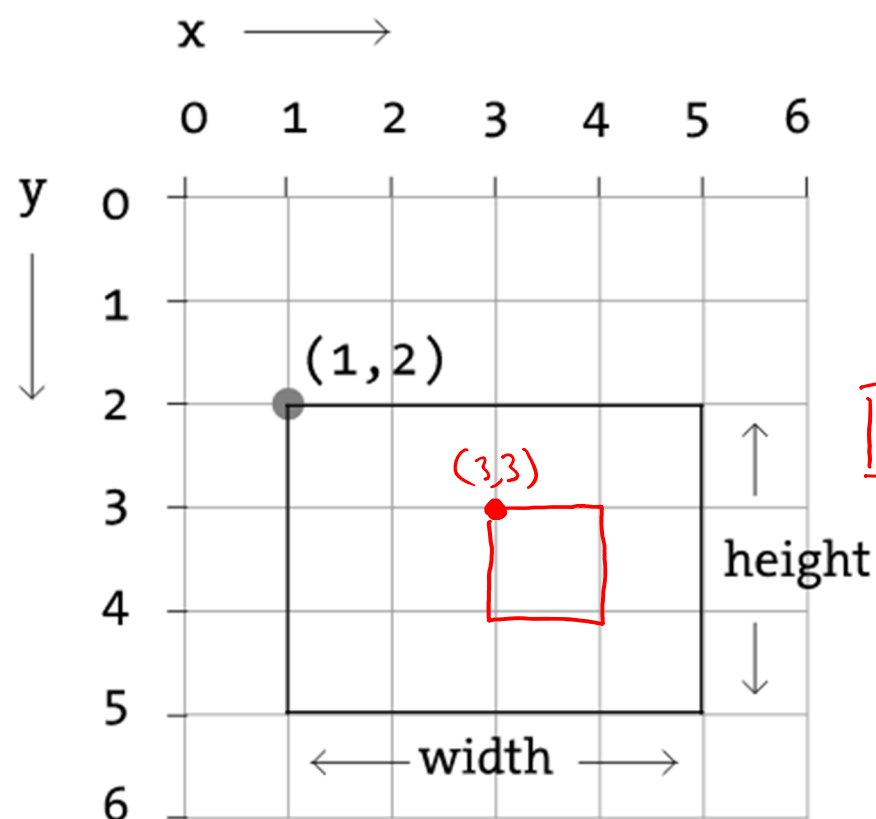
Drawing: Line



Example: `line (1, 2, 5, 2) ;`
`line (5, 5, 2, 3);`

Drawing: Rectangle

- ❖ Default *mode* is CORNER (upper-left)

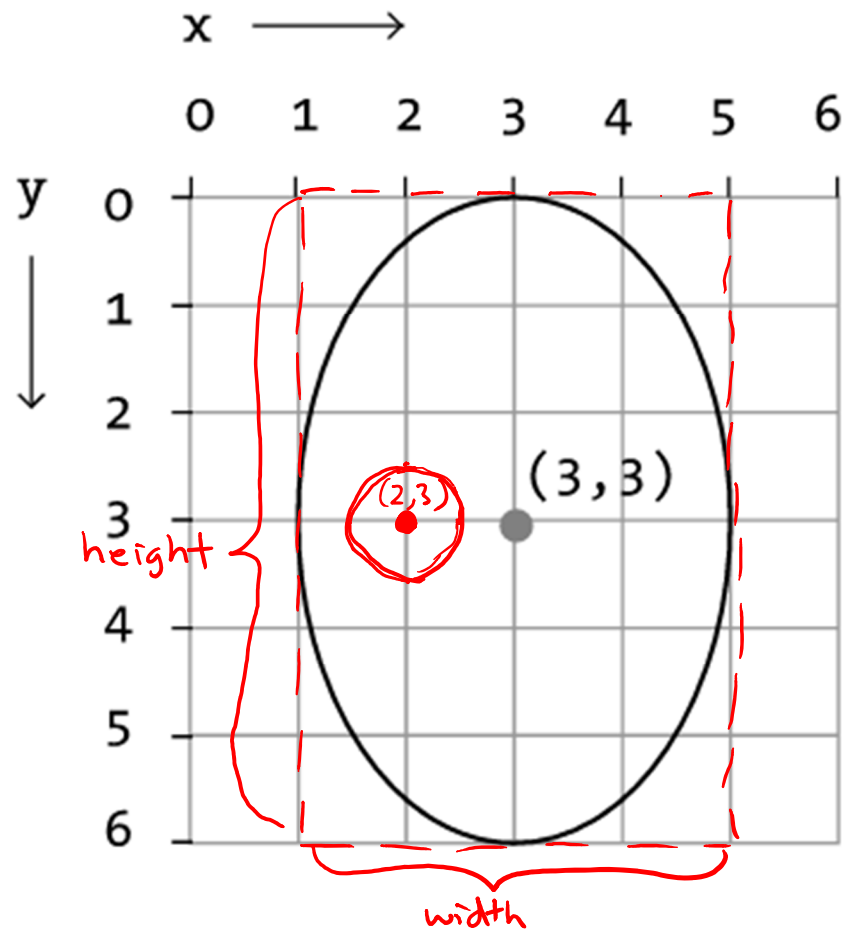


`rect(x, y, width, height);`

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

Drawing: Ellipse/Circle

- ❖ Default *mode* is CENTER



`ellipse(x, y, width, height);`

Example: `ellipse(3, 3, 4, 6);`
`ellipse(2, 3, 1, 1);`

Comments Are Critical!!!

block (multi-line) comment

```
line_drawing
1 /* line_drawing.pde
2    Edited by Justin Hsia (orig. Larry Synder)
3
4    Draws a line to mouse position when user presses mouse.
5 */
6
7 // setup() is a function that runs once at beginning of program
8 void setup() {
9     size(500,500);           // set drawing canvas size to 500x500
10    background(200,200,255); // sets background color to light blue
11 }
12
13 // draw() is a function that runs continuously over and over again
14 void draw() {
15     if(mousePressed) {     // if user presses the mouse
16         stroke(255, 255, 255); // set line color to white
17         line(150, 150, mouseX, mouseY); // draw line from (150,150) to mouse position
18     }
19 }
```

← file name
← your name

← brief program description

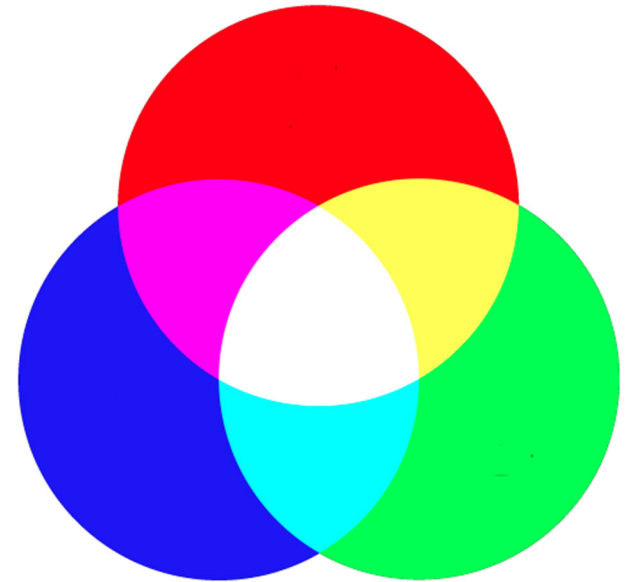
← brief function description

↑ statement description

↑ single-line comment

Understanding Color

- ❖ In electronic systems, color specified using the **RGB color model**
 - Red, Green, Blue
- ❖ 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



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

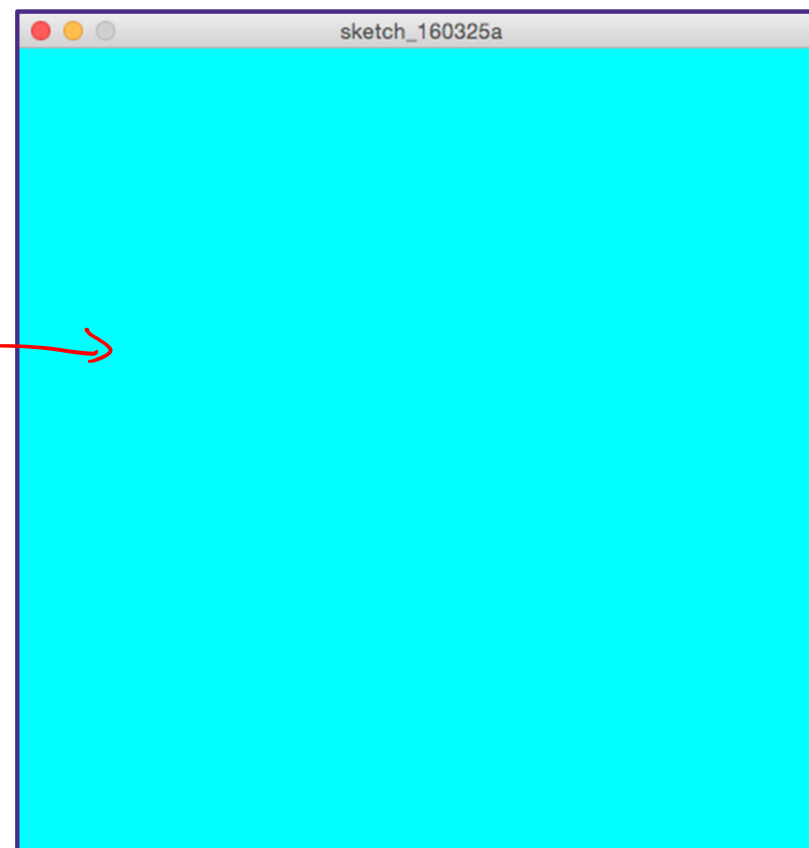
```
❖ color ( R, G, B );  
❖ color (255, 0, 0); // red  
❖ color ( 0, 255, 0); // green  
❖ color ( 0, 0, 255); // blue  
❖ color ( 0, 0, 0); // black  
❖ color (255, 255, 255); // white  
❖ color (255, 255, 0); // yellow  
❖ color (255, 0, 255); // magenta  
❖ color ( 0, 255, 255); // cyan
```

Color Functions

- ❖ `background(R, G, B);`
 - Covers the entire drawing canvas with the specified color
 - Will draw over anything that was previously drawn

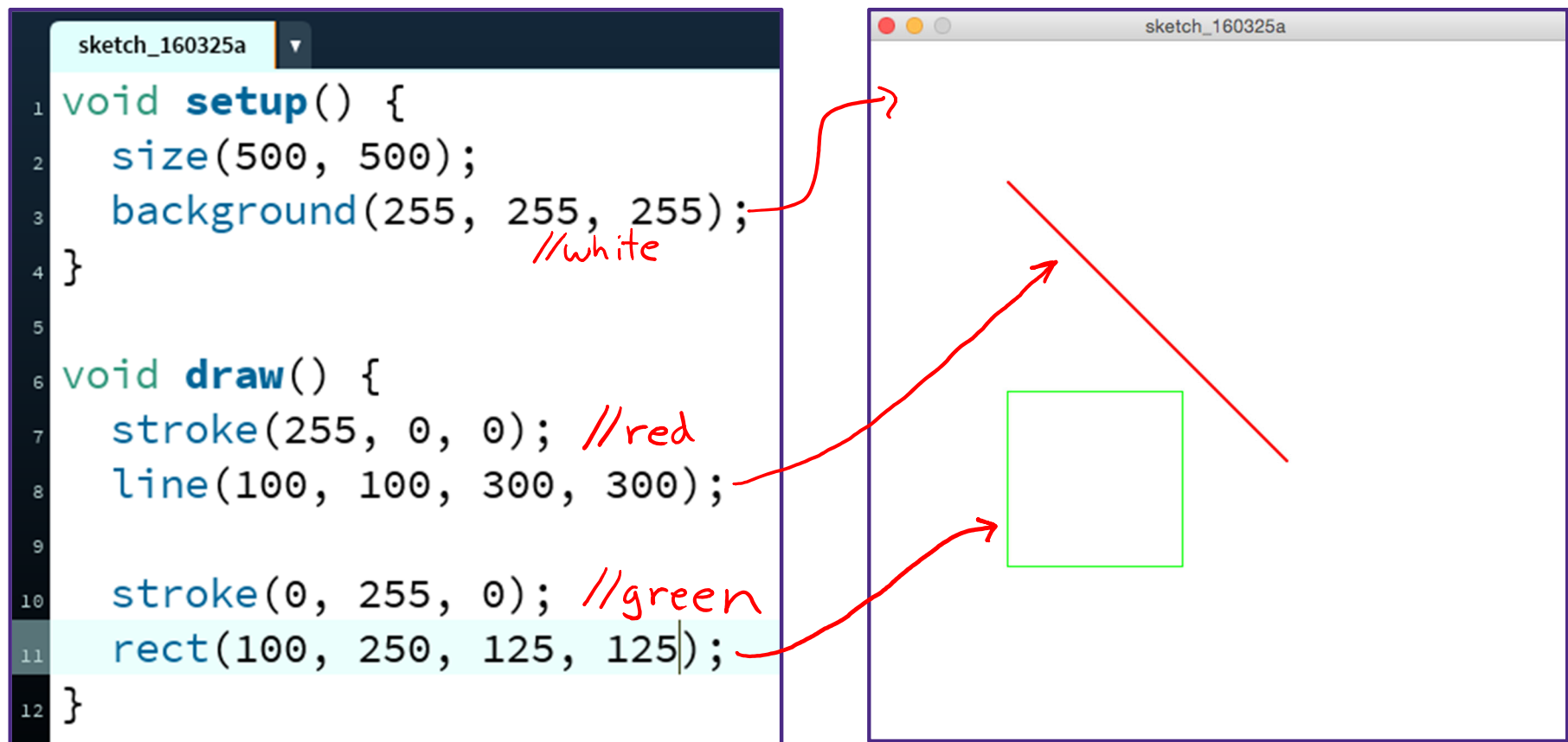
```
sketch_160325a
1 void setup() {
2   size(500, 500);
3   background(0, 255, 255);
4 }
```

Cyan →



Color Functions

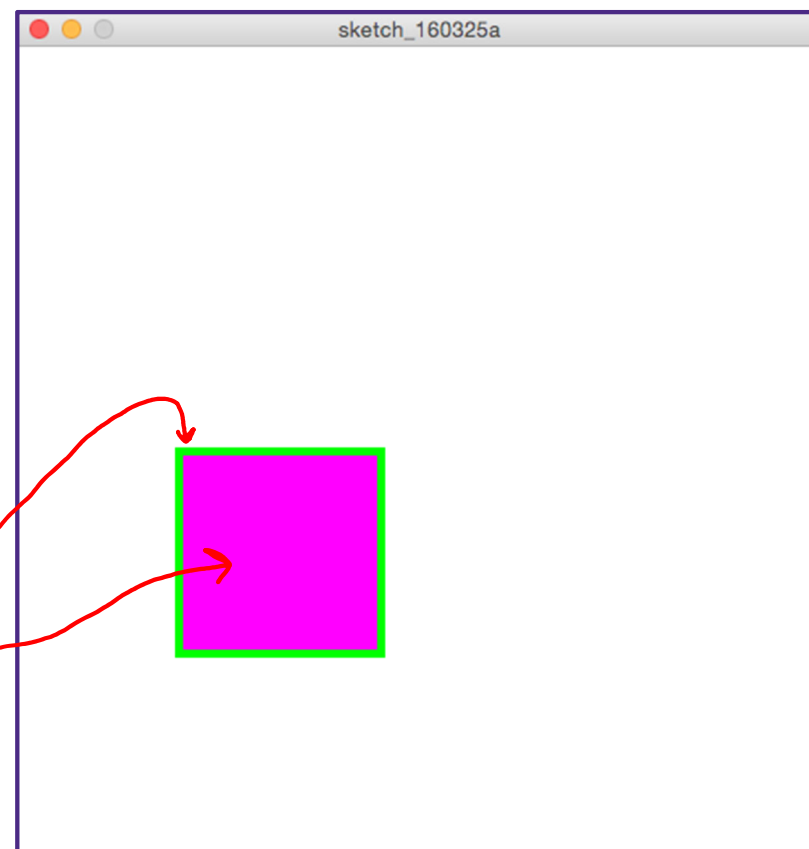
- ❖ `stroke(R, G, B);`
 - Sets the color of the stroke of a *line* or *line around a shape*
 - Can change line size using `strokeWeight(#);`



Color Functions

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

```
sketch_160325a
1 void setup() {
2   size(500, 500);
3   background(255, 255, 255);
4 }
5
6 void draw() {
7   strokeWeight(5);
8   stroke(0, 255, 0); //green
9   fill(255, 0, 255); //magenta
10  rect(100, 250, 125, 125);
11 }
```



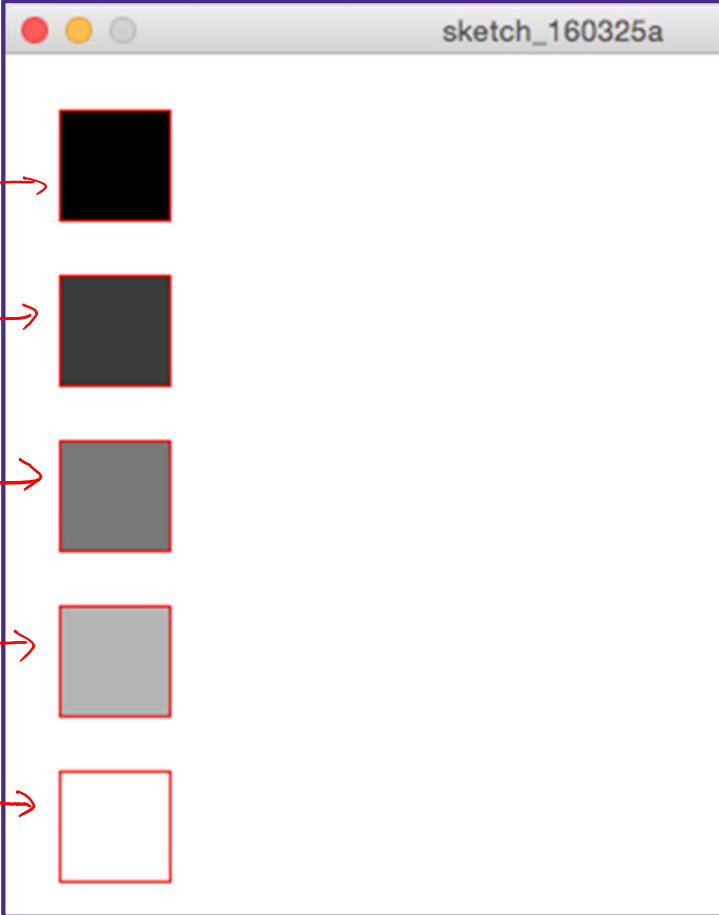
Color: "Grays"

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

darker
(closer to black)

lighter
(closer to white)

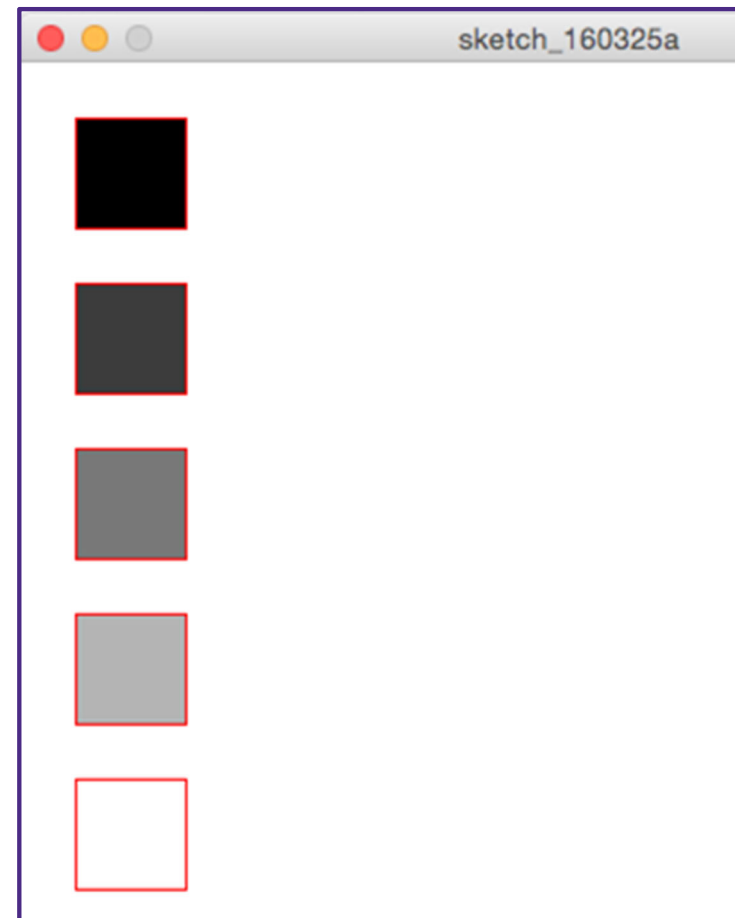
```
6 void draw() {  
7   stroke(255, 0, 0);  
8  
9   fill(0, 0, 0);  
10  rect(25, 25, 50, 50);  
11  
12  fill(60, 60, 60);  
13  rect(25, 100, 50, 50);  
14  
15  fill(120, 120, 120);  
16  rect(25, 175, 50, 50);  
17  
18  fill(180, 180, 180);  
19  rect(25, 250, 50, 50);  
20  
21  fill(255, 255, 255);  
22  rect(25, 325, 50, 50);  
23 }
```



Color: "Grays"

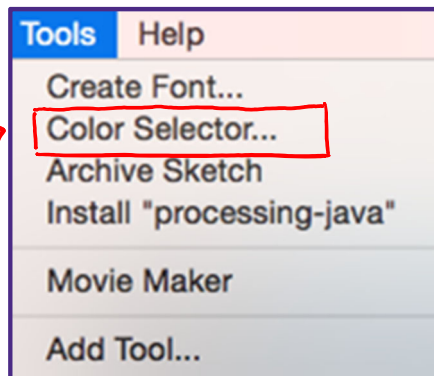
- ❖ When the values for RGB are all the same, then the color will be white, black, or some shade of gray
 - For brevity, can specify just a single number instead

```
6 void draw() {  
7   stroke(255, 0, 0);  
8  
9   fill(0);  
10  rect(25, 25, 50, 50);  
11  
12  fill(60);  
13  rect(25, 100, 50, 50);  
14  
15  fill(120);  
16  rect(25, 175, 50, 50);  
17  
18  fill(180);  
19  rect(25, 250, 50, 50);  
20  
21  fill(255);  
22  rect(25, 325, 50, 50);  
23 }
```



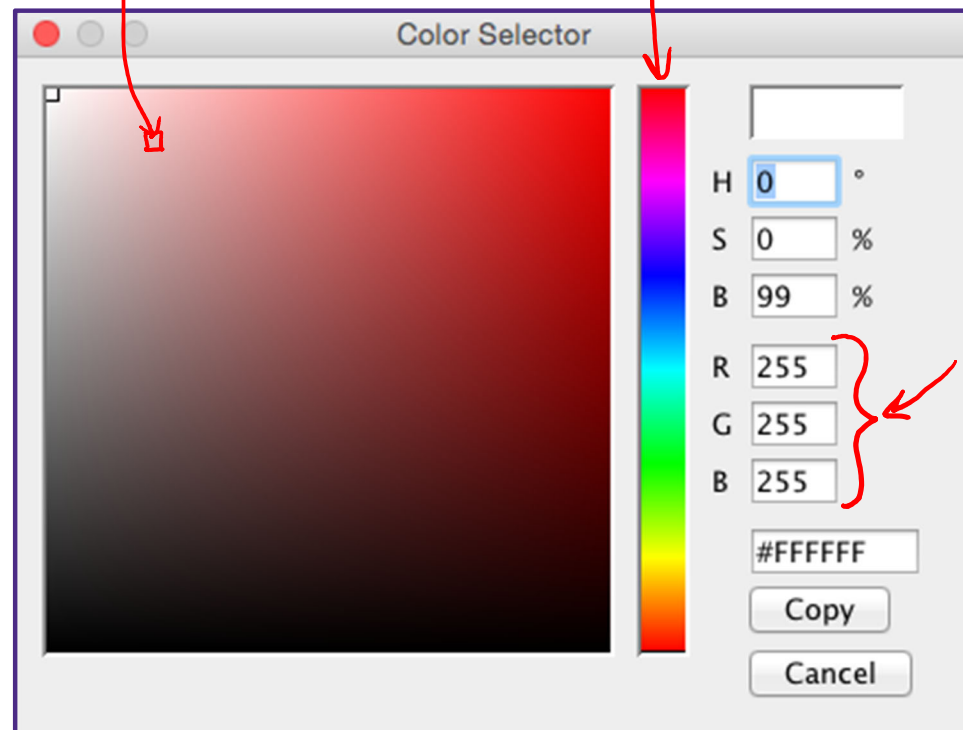
Processing's Color Selector

⑥ open color selector



② use color field to select color

① use color slider to get to different color ranges



③ copy RGB values from here

Color Selector

H °

S %

B %

R

G

B

#FFFFFF

Copy

Cancel

The Color “State” of Your Program

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

Practice 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 (line)  
fill(183, 165, 122);         // UW gold (inside)  
ellipse(100, 100, 100, 200);  
                                taller
```

A.



~~B.~~



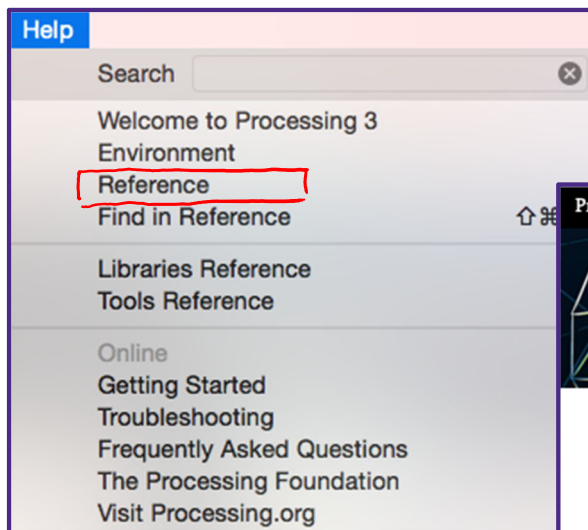
C.



~~D.~~



The Processing Reference



Processing

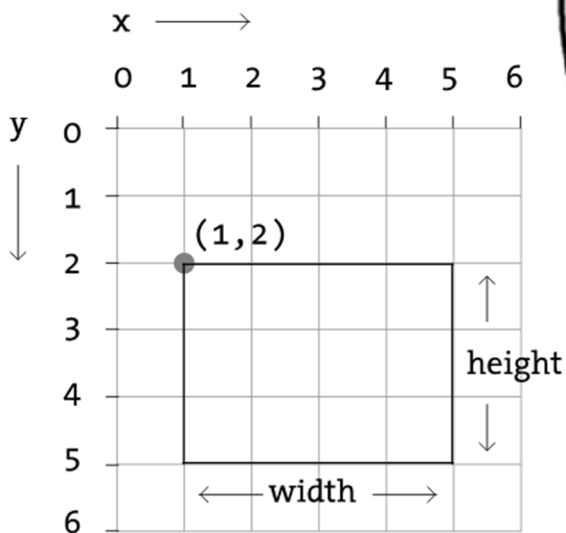
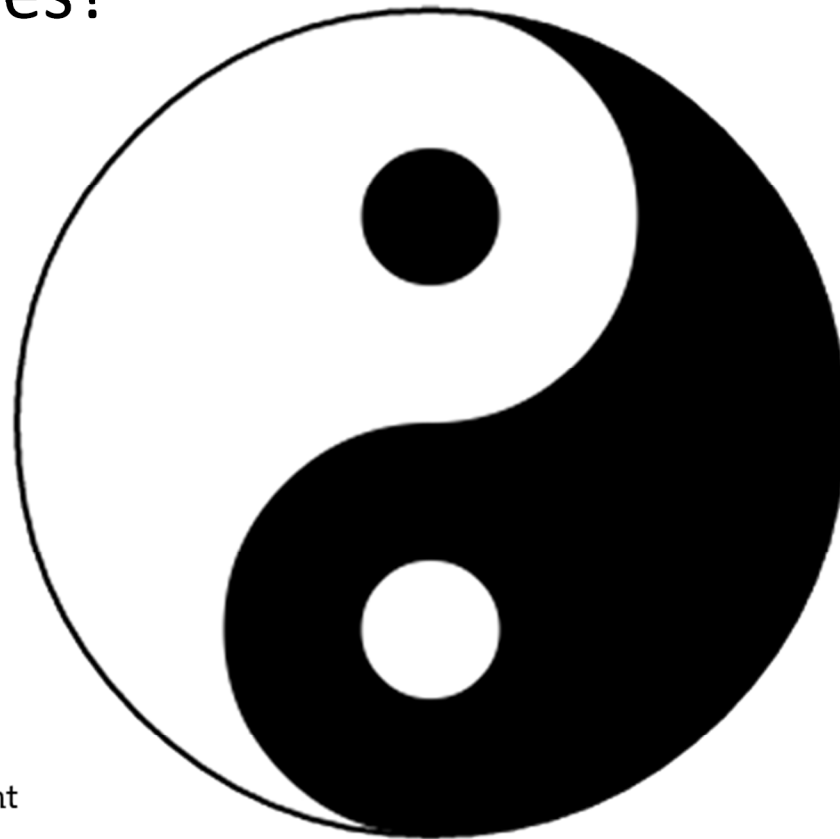
Reference. Processing was designed to be a flexible software sketchbook.

Language
Libraries
Tools
Environment

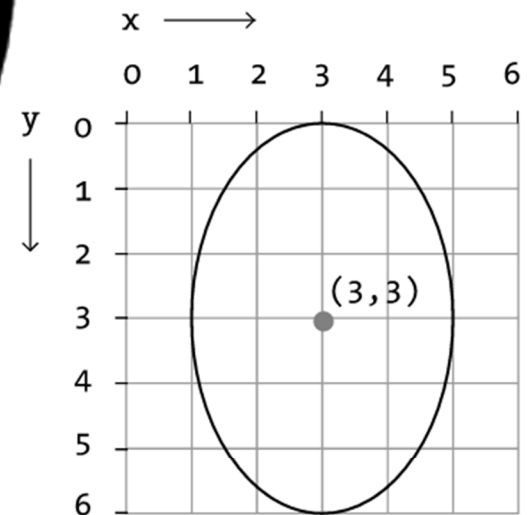
<p>Structure</p> <ul style="list-style-type: none"> () (parentheses) , (comma) . (dot) /* */ (multiline comment) /** */ (doc comment) // (comment) ;(semicolon) = (assign) [] (array access) { } (curly braces) catch class draw() exit() extends false final implements import loop() new noLoop() null popStyle() 	<p>Shape</p> <ul style="list-style-type: none"> createShape() loadShape() PShape <p>2D Primitives</p> <ul style="list-style-type: none"> arc() ellipse() line() point() quad() rect() triangle() <p>Curves</p> <ul style="list-style-type: none"> bezier() bezierDetail() bezierPoint() bezierTangent() curve() curveDetail() curvePoint() curveTangent() curveTightness() 	<p>Color</p> <p>Setting</p> <ul style="list-style-type: none"> background() clear() colorMode() fill() noFill() noStroke() stroke() <p>Creating & Reading</p> <ul style="list-style-type: none"> alpha() blue() brightness() color() green() hue() lerpColor() red() saturation() <p>Image</p> <ul style="list-style-type: none"> createImage()
--	--	--

Activity: Taijitu

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



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



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