Processing and Drawing

CSE 120 Winter 2020

Instructor: Sam Wolfson

Teaching Assistants: Yae Kubota Eunia Lee Erika Wolfe

'Do Not Sell My Info': U.S. retailers rush to comply with California privacy law

Large U.S retailers are rushing to comply with a new law, the California Consumer Privacy Act (CCPA), which becomes effective at the start of 2020 and is one of the most significant regulations overseeing the data collection practices of U.S. companies. It lets shoppers opt out of allowing retailers and other companies to sell personal data to third parties.

In addition to retailers, the law affects a broad swath of firms including social media platforms such as Facebook and Alphabet's Google, advertisers, app developers, mobile service providers and streaming TV services, and is likely to overhaul the way companies benefit from the use of personal information.

The law follows Europe's controversial General Data Protection Regulation, which set a new standard for how companies collect, store and use personal data. The European law gave companies years to comply while CCPA has given them a few months.

- https://www.reuters.com/article/us-usa-retail-privacy/do-not-sell-my-info-u-s-retailers-rush-to-comply-with-california-privacy-law-idUSKBN1YY0RK

Administrivia

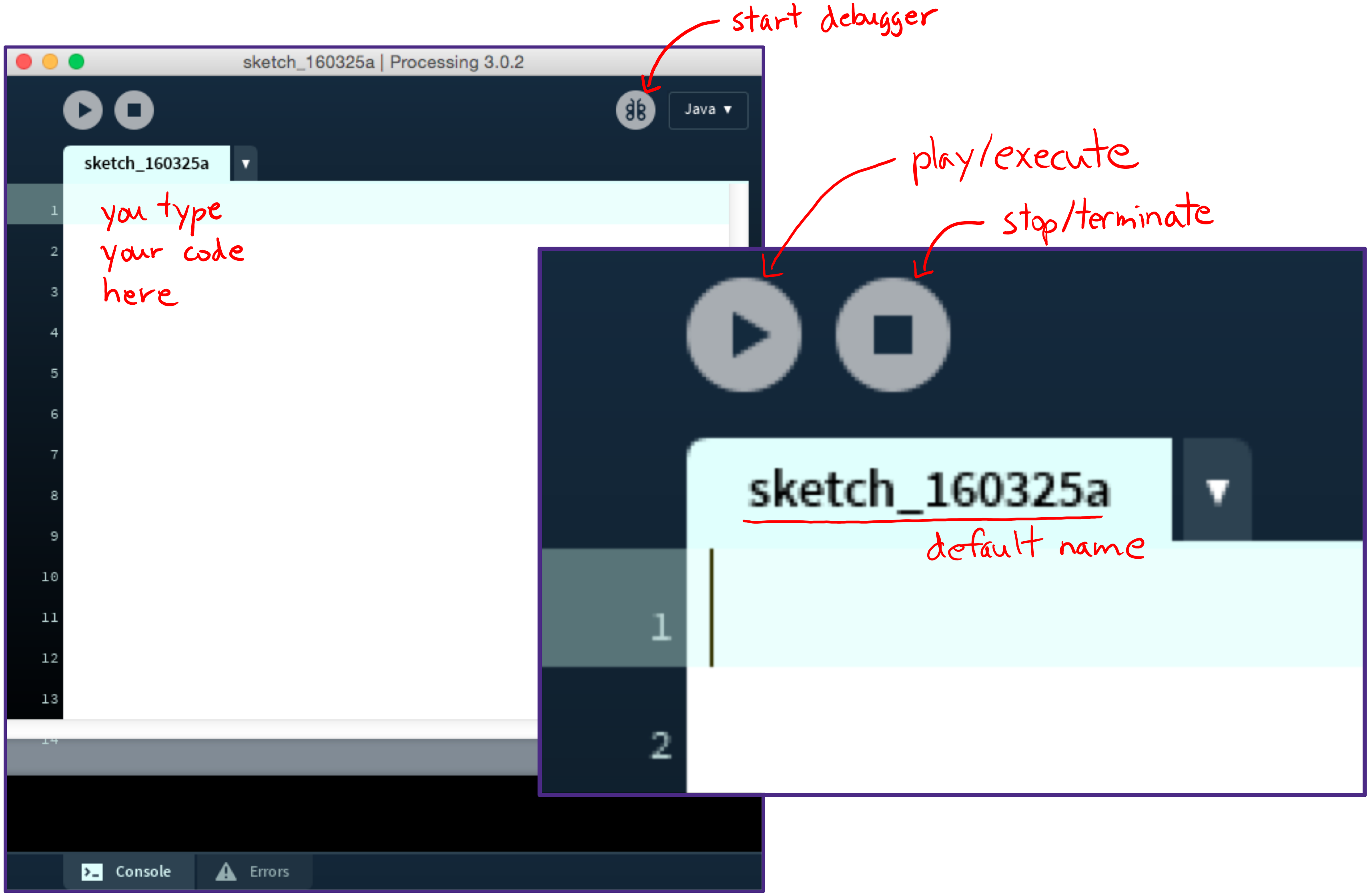
- Assignments:
  - Lightbot Functions [hw] due today *before* 11:59 pm (1/13)
  - Taijitu [lab] due by end of Thursday (1/16)
- "Big Ideas" lecture this week: Algorithms
  - Reading due before lab on Thursday (1/16)
- Register on Piazza (8 of you still haven't)
- Grading and Grades
  - Reading Check 1 and Personal Values scores released soon
  - 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

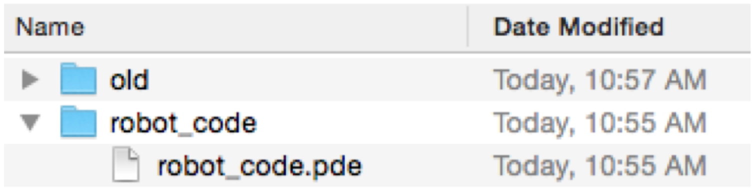
- It is both a programming *environment* (where you type) and a programming *language*
  - You are writing Java code, but they have made a lot of things easier

The Processing Coding Environment

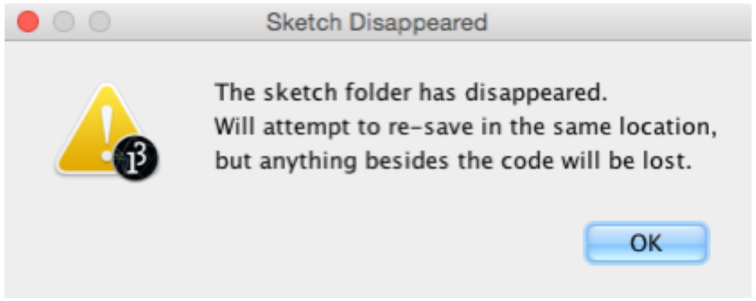


Aside: Processing Files

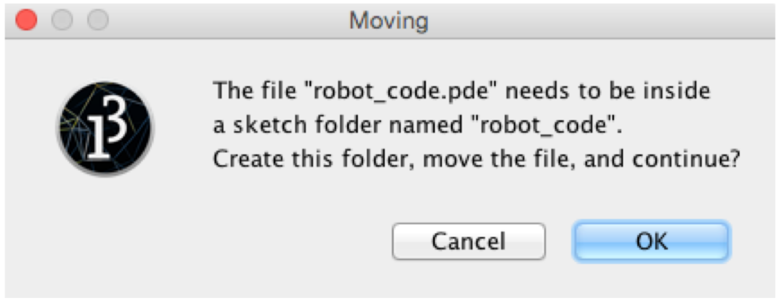
- Processing files have extension `.pde`
  - File names *cannot* contain dashes (−) use underscore (_) instead
- To run a Processing file, it *must* be in a folder of the same name
  - If it's not, then Processing will create the folder for you



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

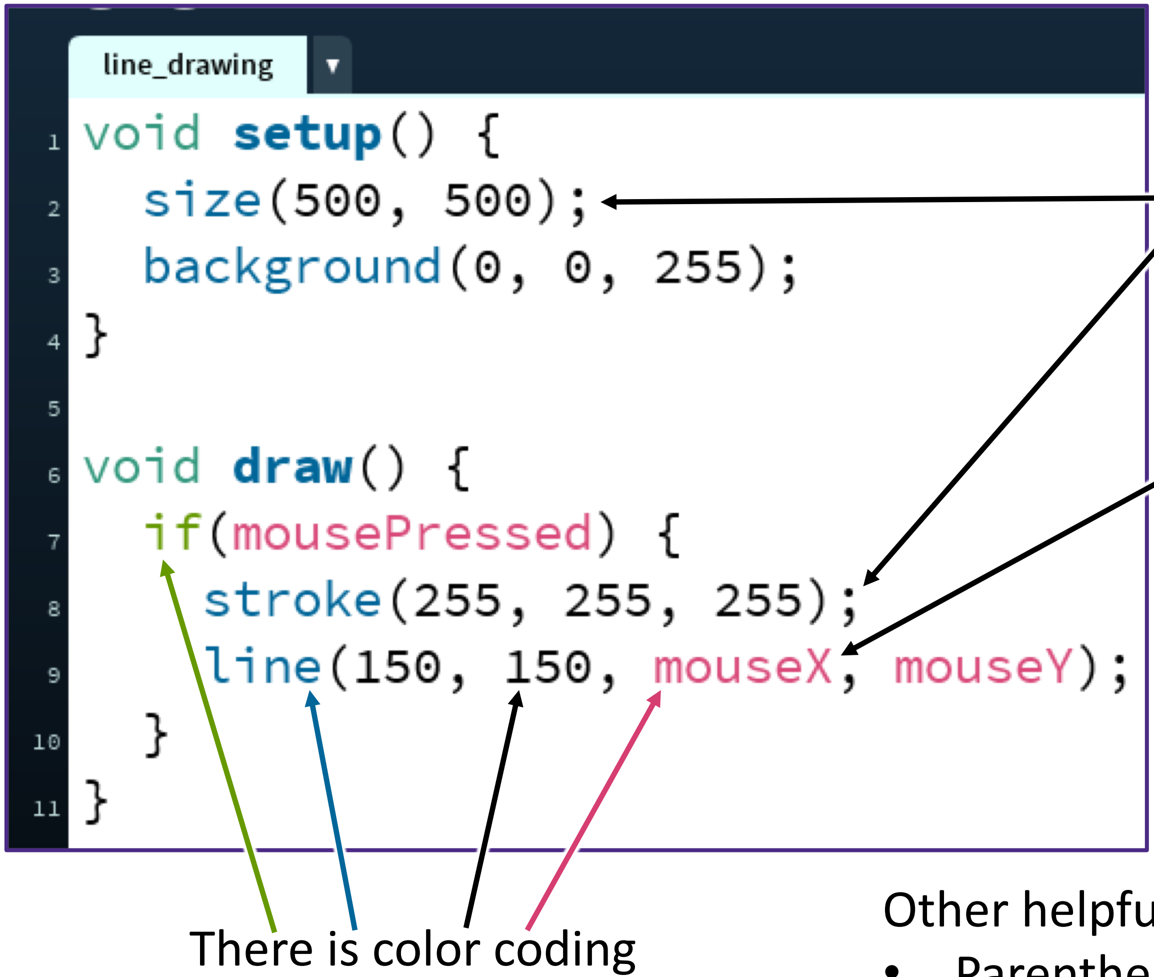


The sketch folder has disappeared.
Will attempt to re-save in the same location,
but anything besides the code will be lost.



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

Text-Based Programming Basics



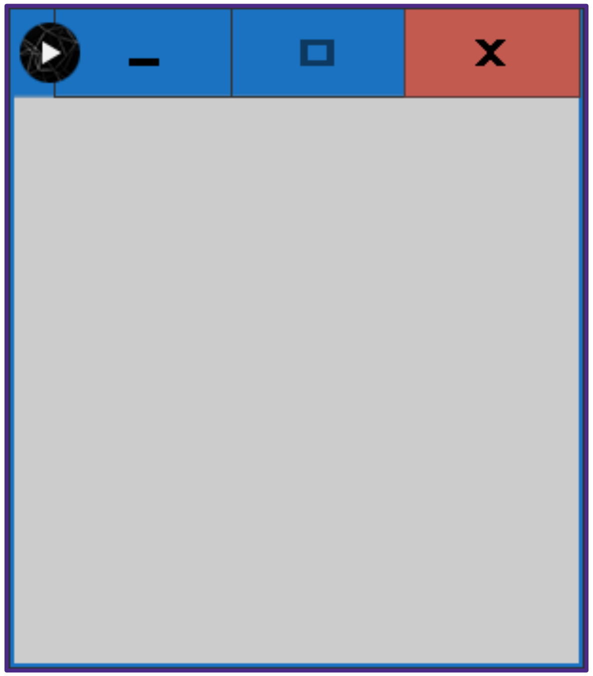
```
void setup() {
  size(500, 500);
  background(0, 0, 255);
}

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

Other helpful *environment* features:

- Parentheses matching
- Error messages

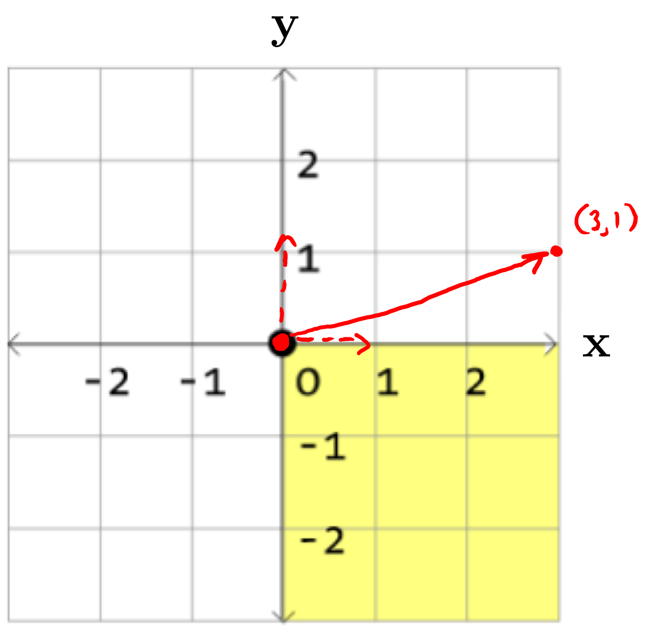
The Drawing Canvas

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



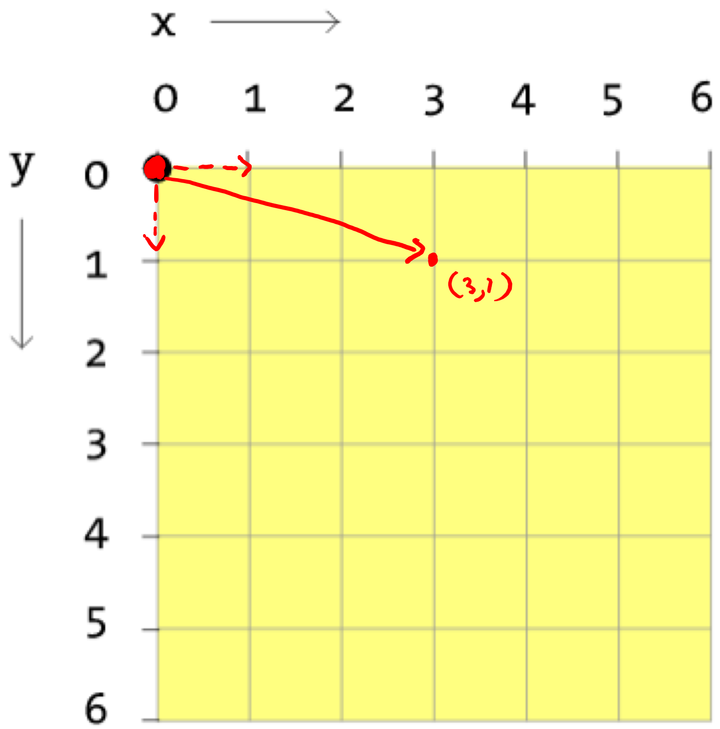
Coordinate System

Math



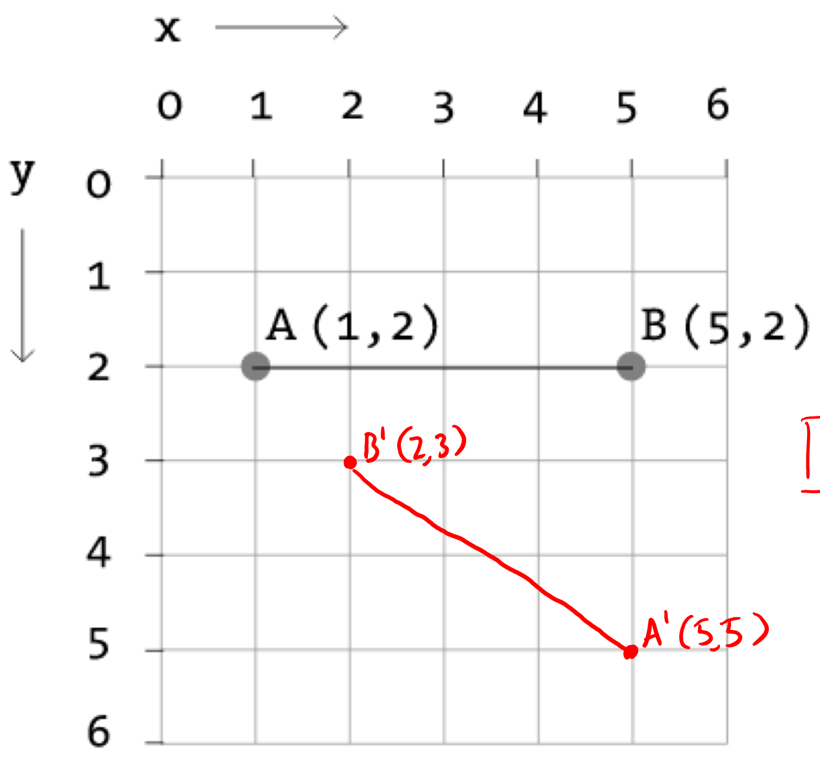
origin (0,0) is center

Processing



origin (0,0) is upper-left

Drawing: Line



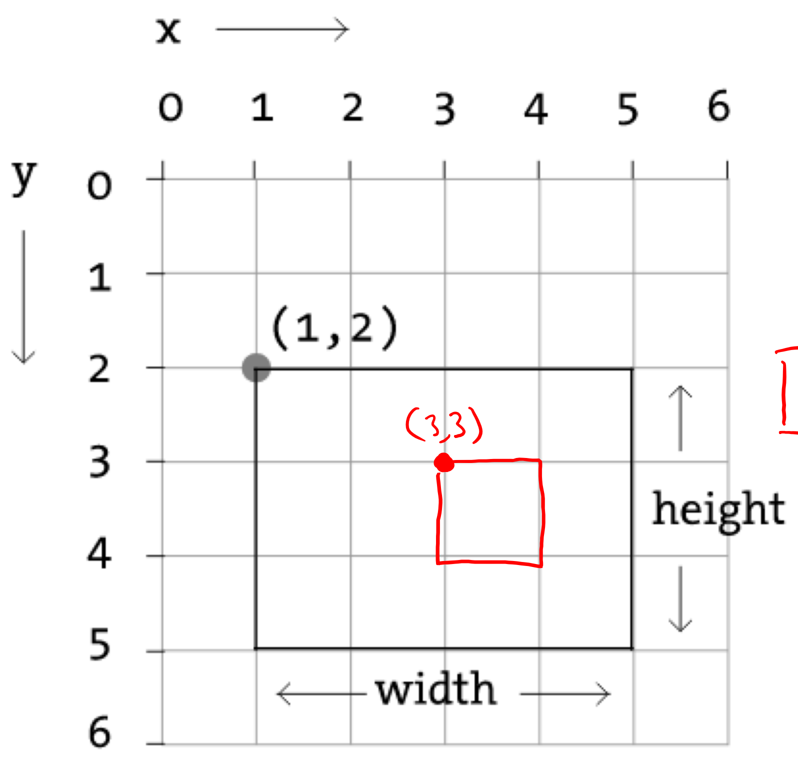
$\boxed{\text{line(x1,y1,x2,y2);}}$

Point A Point B

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

Drawing: Rectangle

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

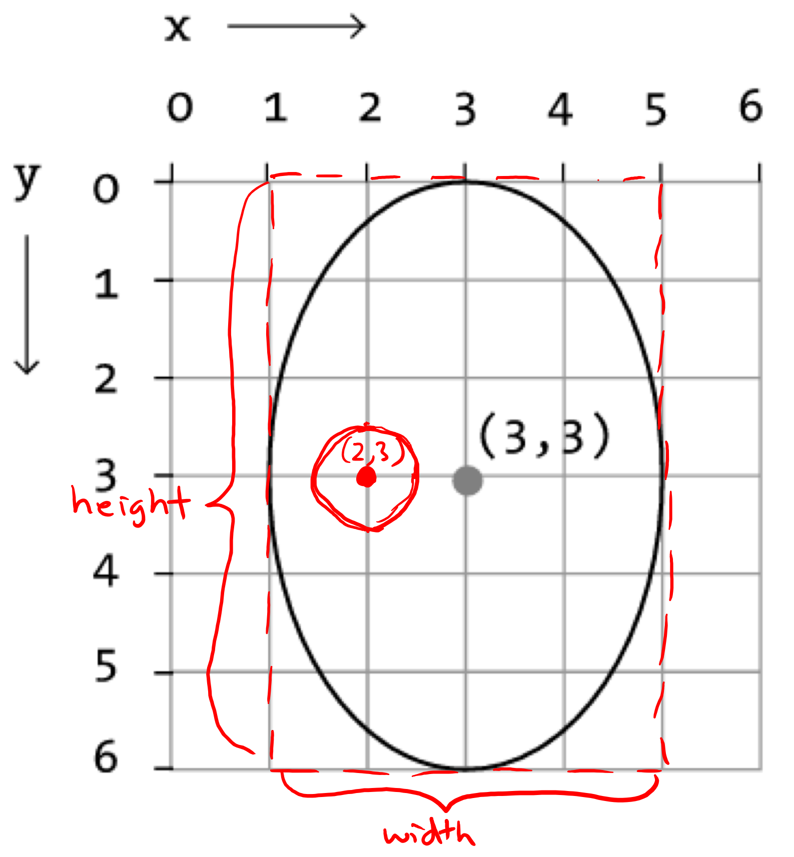


rect (x, y, width, height);

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

Drawing: Ellipse/Circle

- Default *mode* is `CENTER`



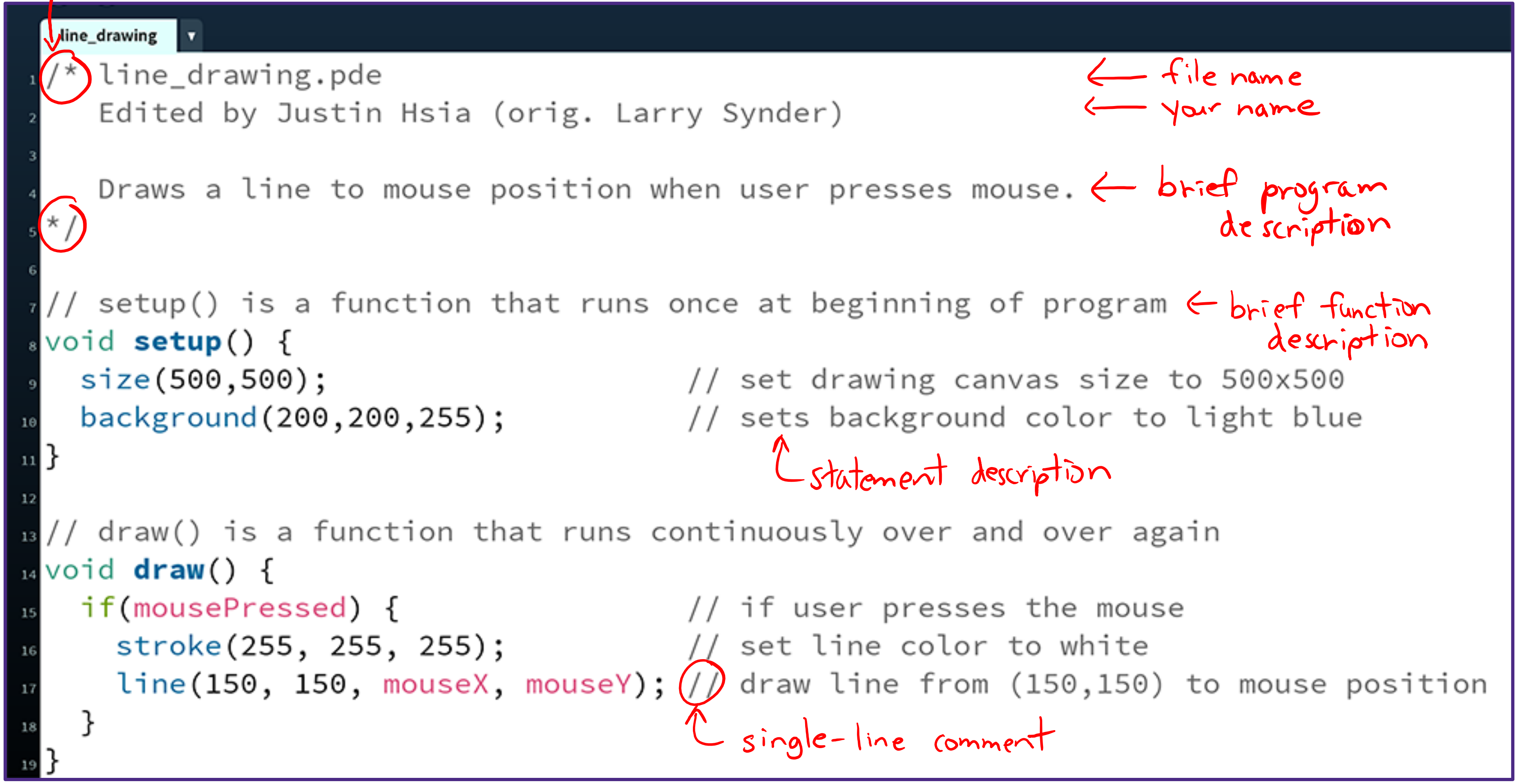
$\boxed{\text{ellipse(x, y, width, height);}}$

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

ellipse(2, 3, 1, 1);

Comments Are Critical!!!

block (multi-line) comment



```
/* line_drawing.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
  }
}
```

← file name

← your name

← brief program description

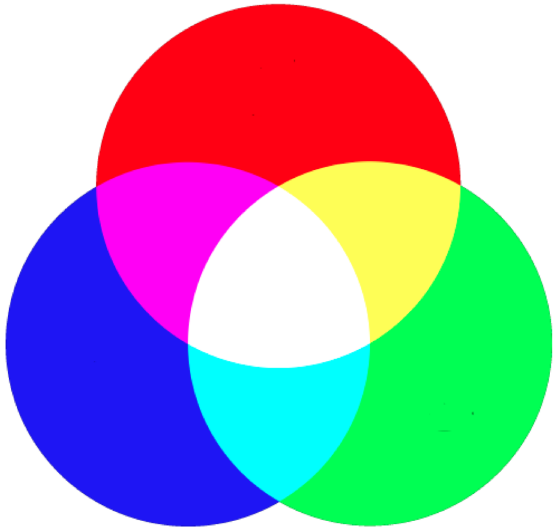
← brief function description

statement description

single-line comment

Understanding Color

- In electronic systems, color specified using the RGB color model
  - **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

Guess the Color

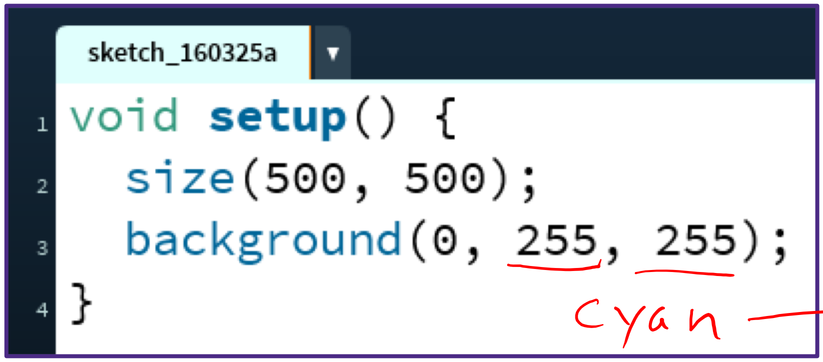
- `color(  R,   G,   B);`
- `color(255,   0,   0); // red`
- `color(  0, 255,   0); // green`
- `color(  0,   0, 255); // blue`
- `color(  0,   0,   0); // black`
- `color(255, 255, 255); // white`
- `color(255, 255,   0); // yellow`
- `color(255,   0, 255); // magenta`
- `color(  0, 255, 255); // cyan`

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

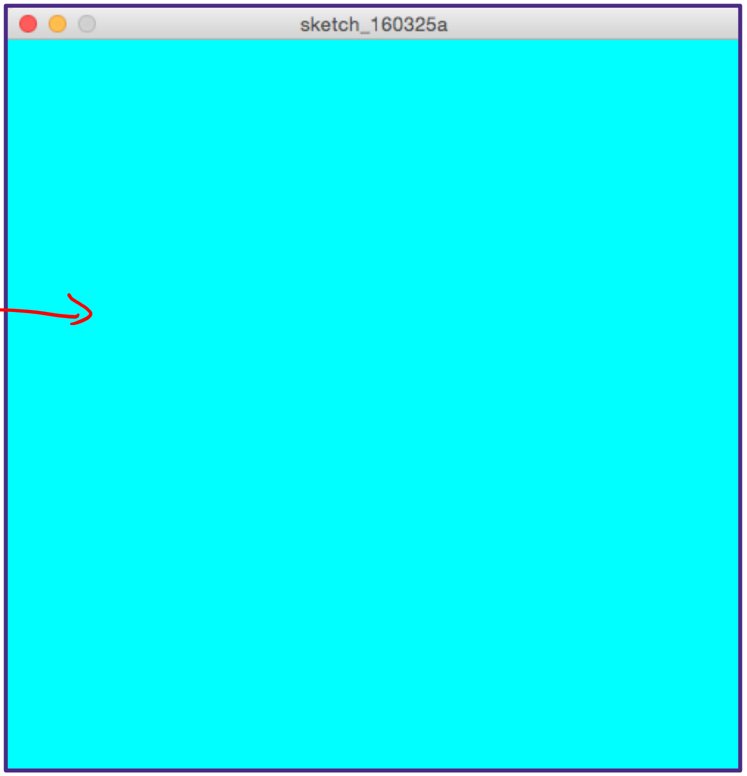
- `background(R, G, B);`
  - Covers the entire drawing canvas with the specified color
  - Will draw over anything that was previously drawn



sketch_160325a

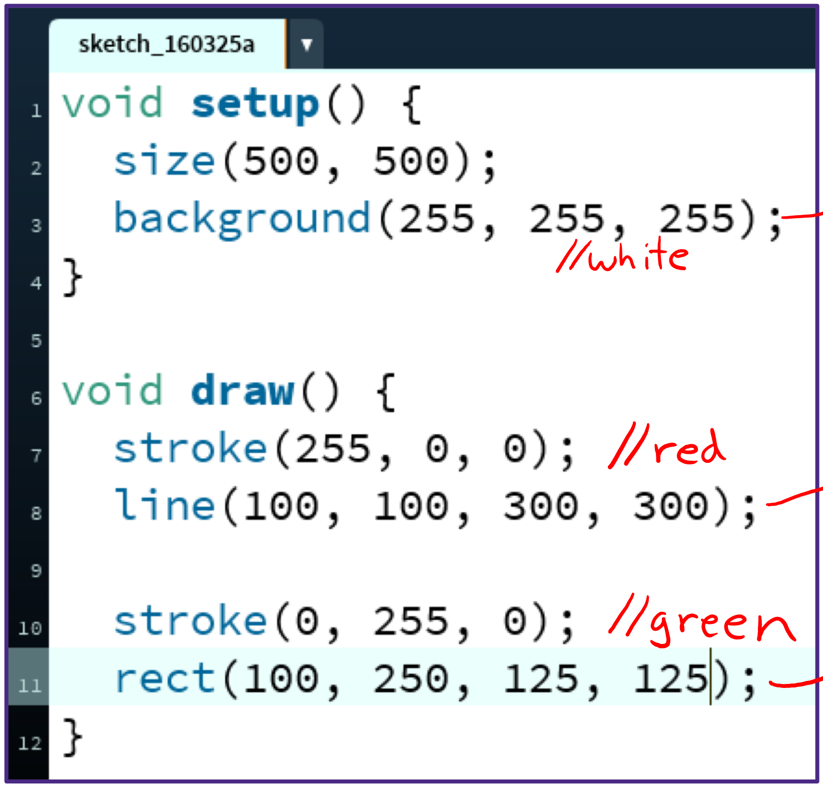
```
void setup() {
  size(500, 500);
  background(0, 255, 255);
}
```

cyan

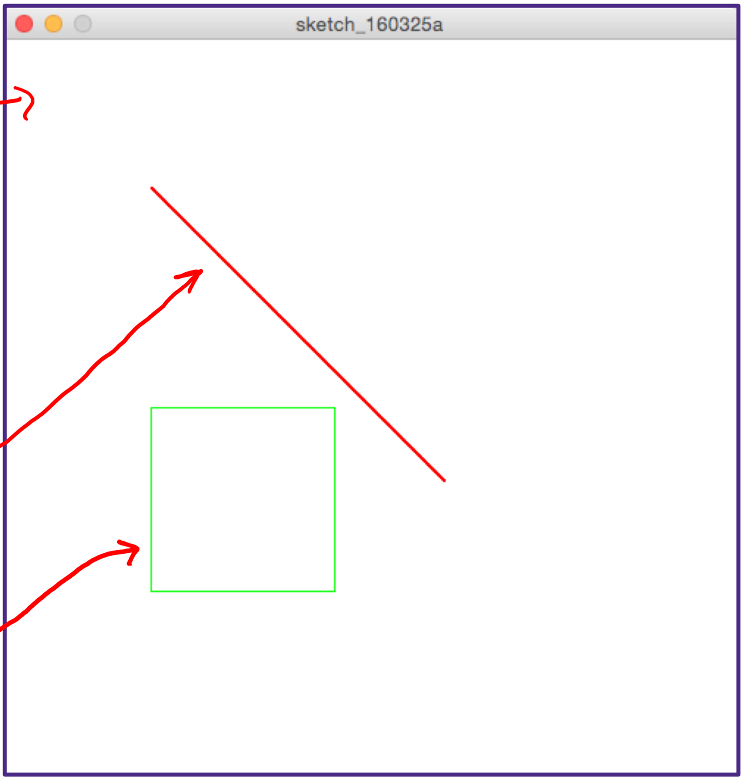


Color Functions

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

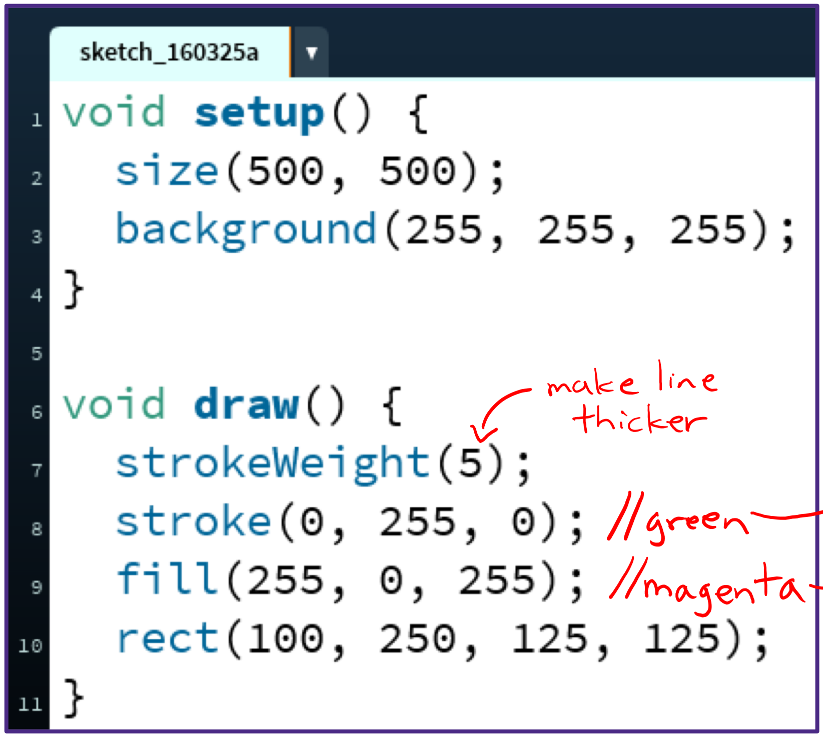


```
void setup() {
  size(500, 500);
  background(255, 255, 255);
                   //white
}

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

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



Color Functions

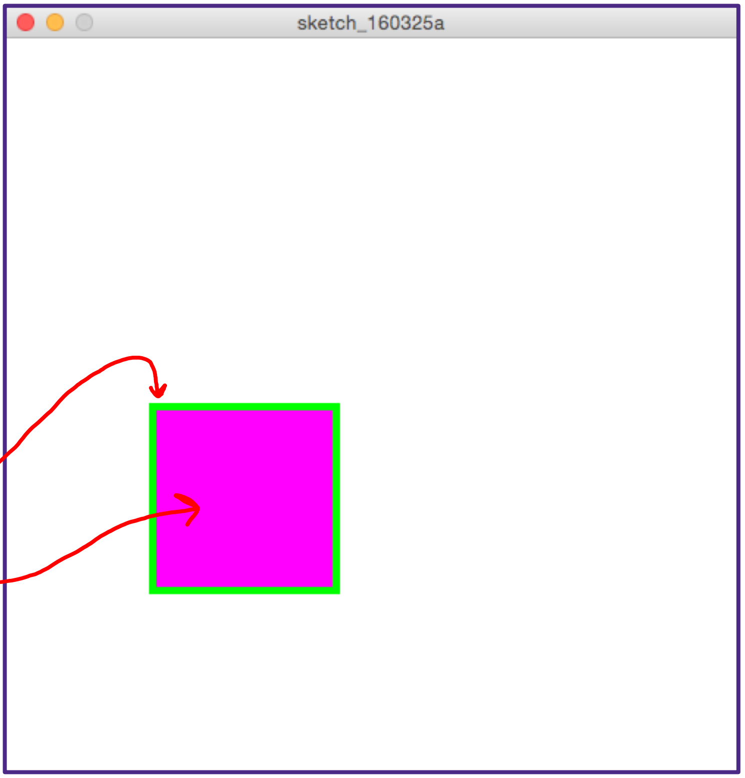
- `fill(R, G, B);`
  - Sets the *inside* color of a shape (**note:** you cannot fill a line)



sketch_160325a

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

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

make line thicker

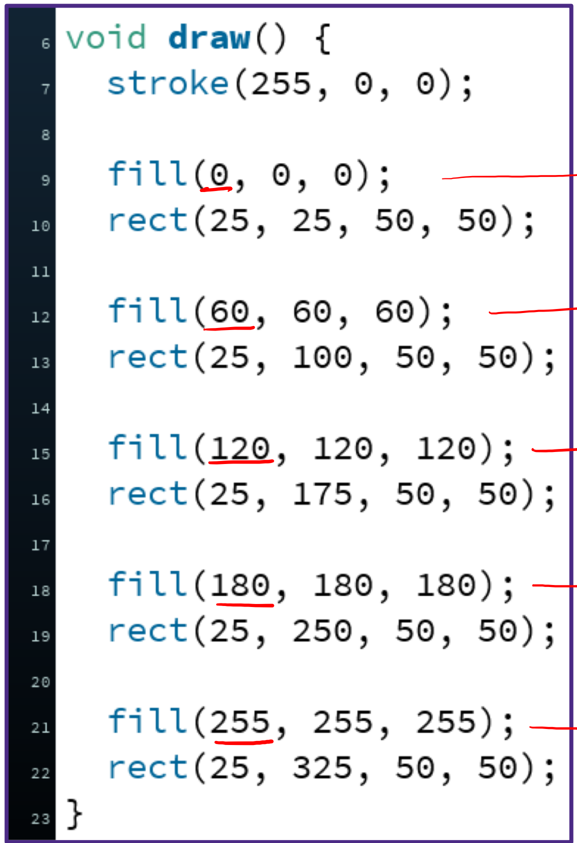


Color: "Grays"

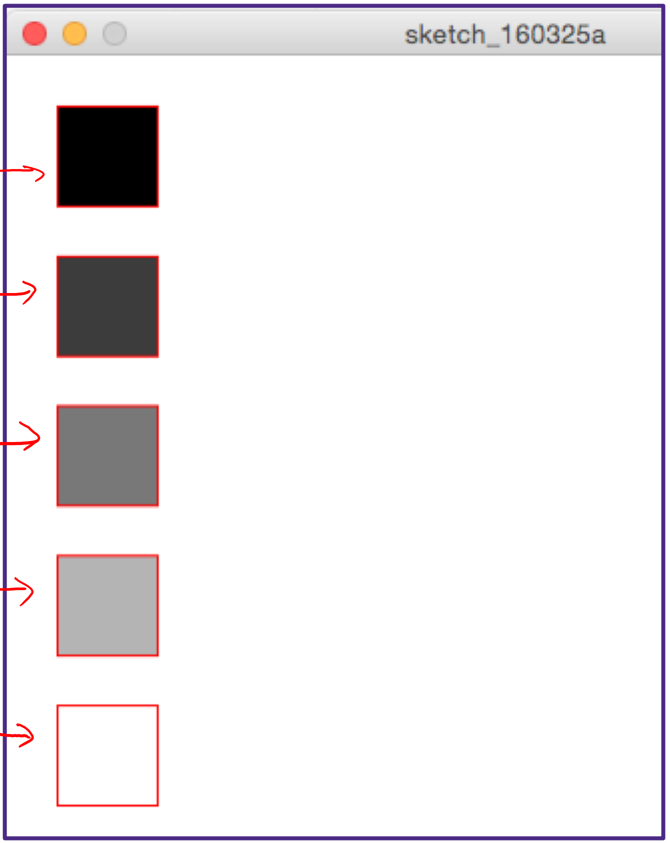
- When the values for RGB are all the same, then the color will be white, black, or some shade of gray

darker (closer to black)

lighter (closer to white)

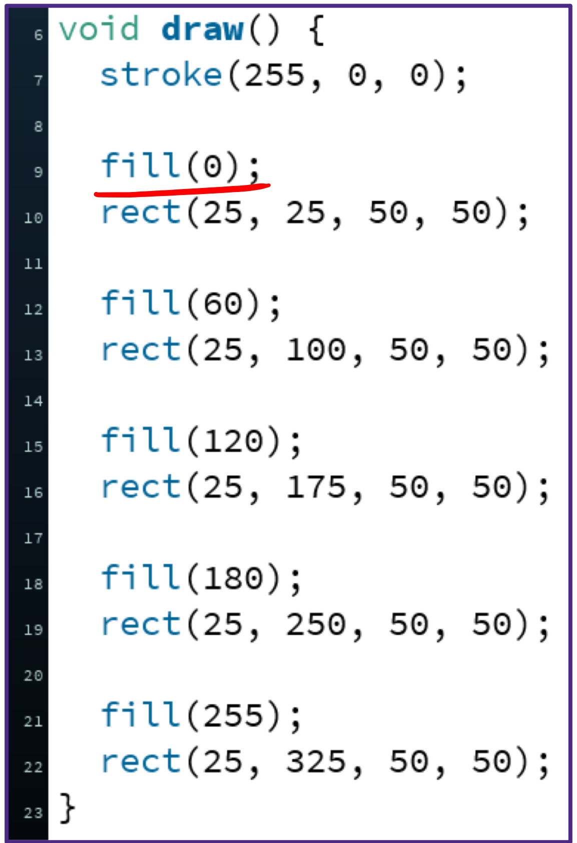


```
void draw() {
  stroke(255, 0, 0);

  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);
}
```

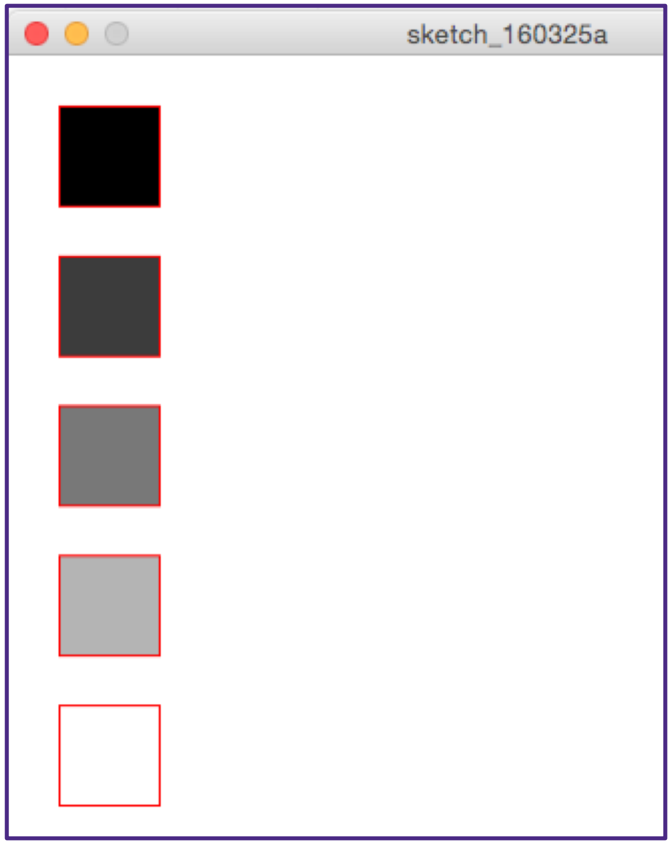


Color: “Grays"

- 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

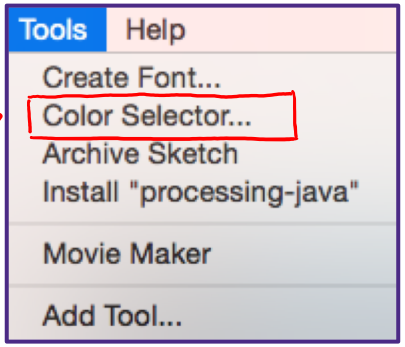


```
void draw() {
  stroke(255, 0, 0);

  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);
}
```



Processing's Color Selector

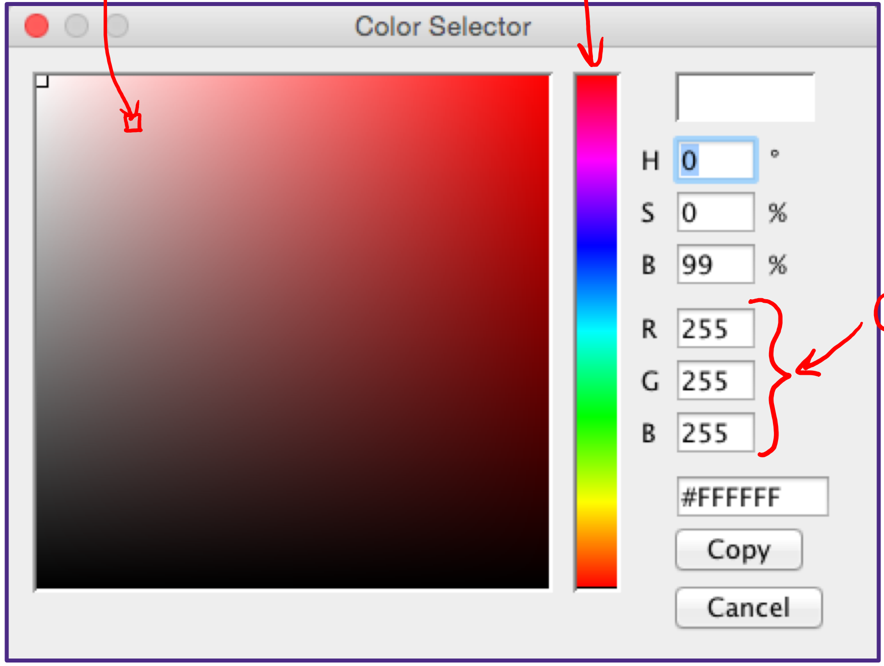
⓪ open color selector



① use color slider to get to different color ranges

② use color field to select color

③ copy RGB values from here



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

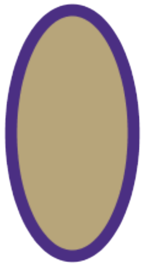
The Color "State" of Your Program

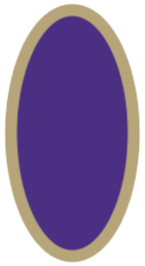
- Recall that programs are executed sequentially (*i.e.* instruction-by-instruction)
- `stroke()` and `fill()` apply to *all* subsequent drawing statements
  - Until a later call overrides
- Hidden color "state" that knows the current values of `stroke()`, `strokeWeight()`, and `fill()`
  - In complex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

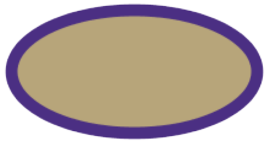
Practice Question

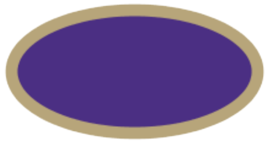
- Which of the following drawings corresponds to the Processing code below?
  - Talk with your neighbors!

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

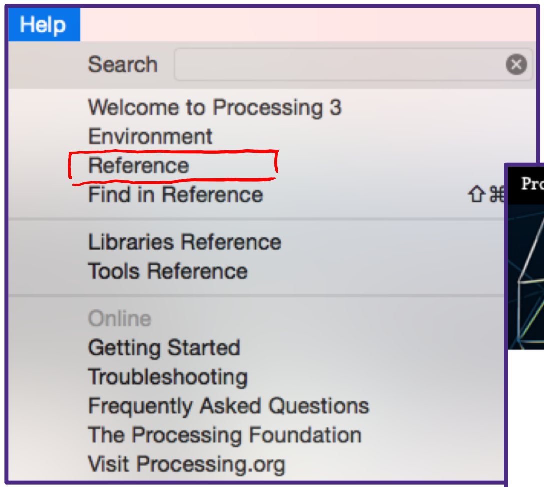
A. 

B. 

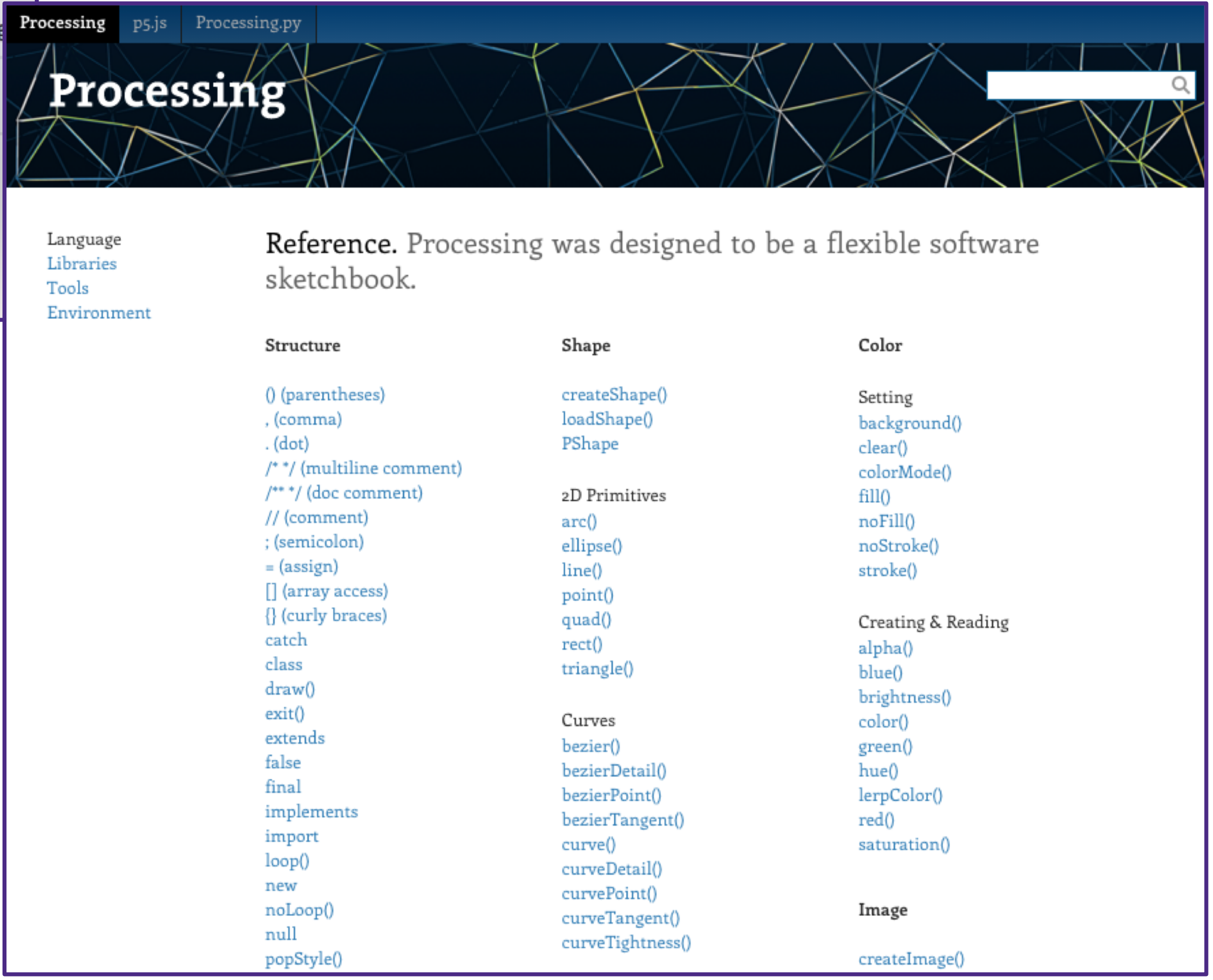
C. 

D. 

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

() (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()

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

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

Color

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

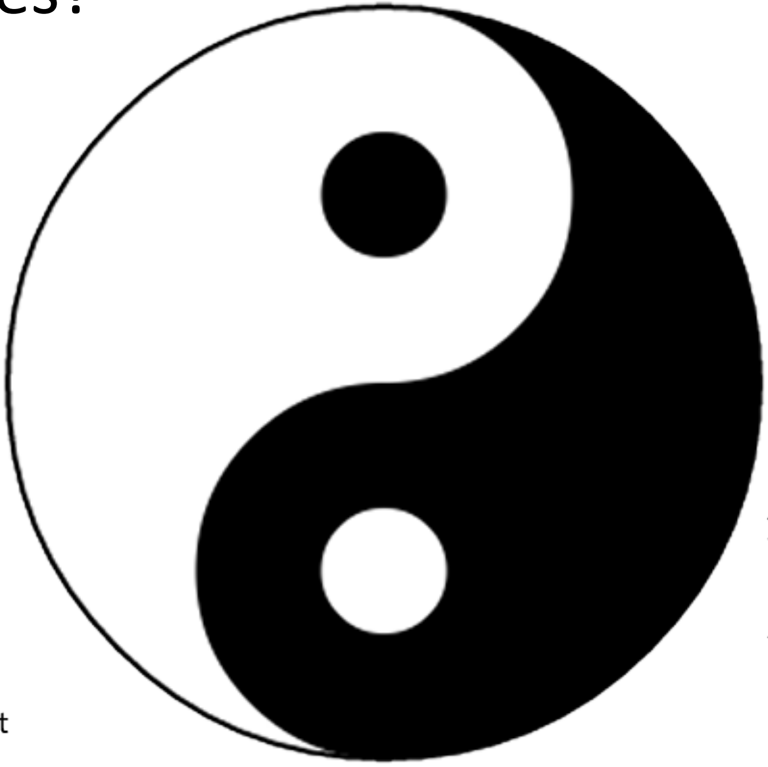
Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

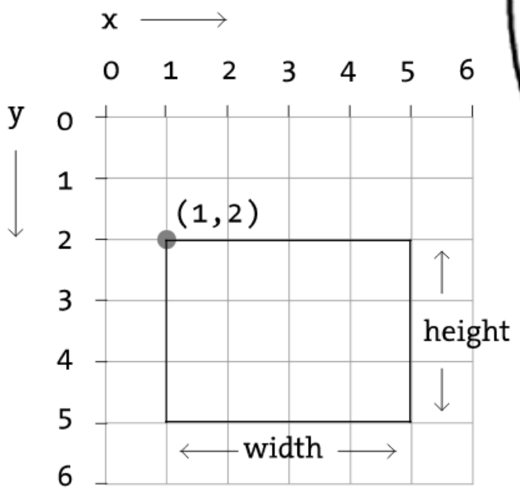
Image

createImage()

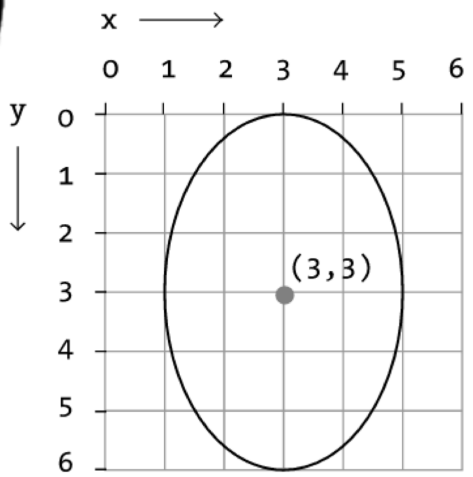
Activity: Taijitu

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





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



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