

WELCOME TO THE 4%



Code Liberation x Indiecade 2014

A trans-inclusive, women-only programming workshop

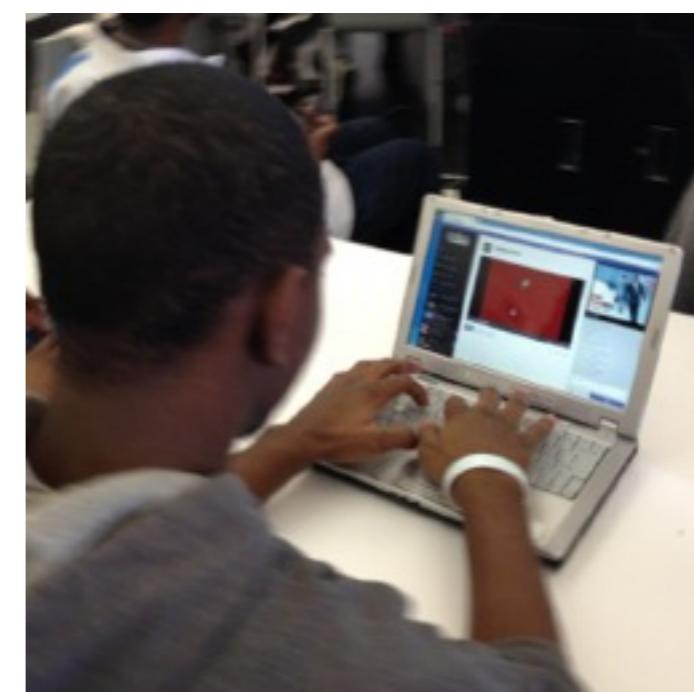
WHO ARE YOU?

WHO AM I?

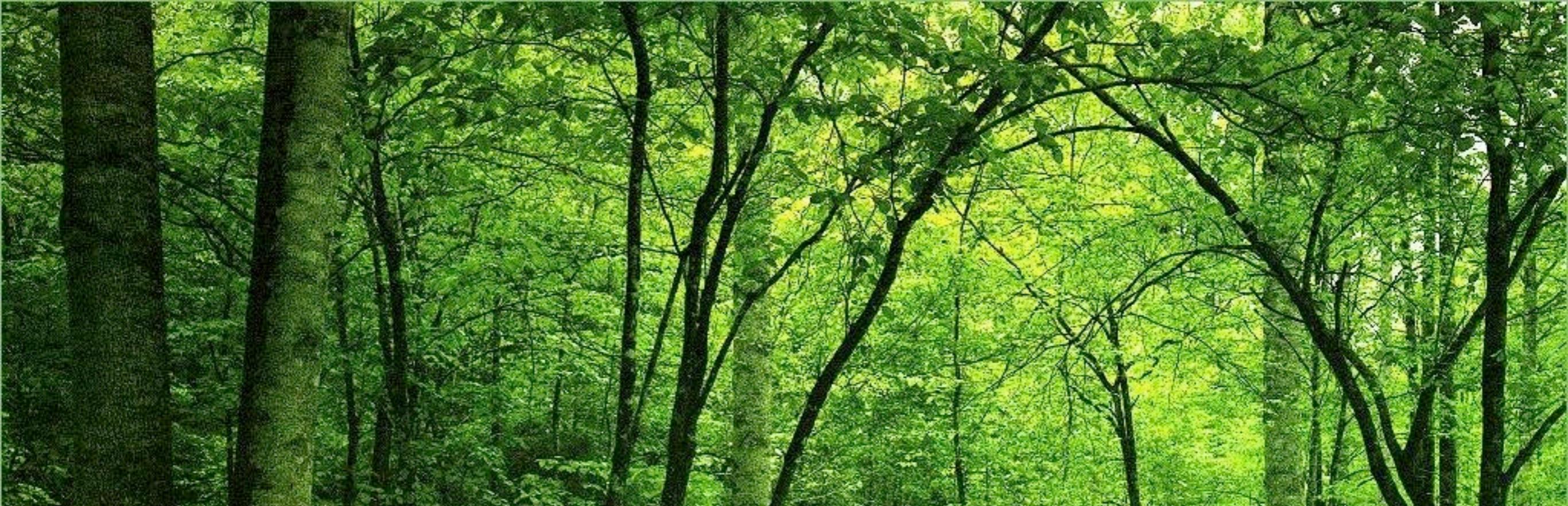


Jane Friedhoff
Creative Coder & Game Designer
janefriedhoff.com
[@jfriedhoff](https://twitter.com/jfriedhoff)

c/o interstellarselfiestation.com



(among others!)



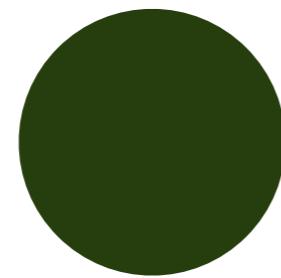
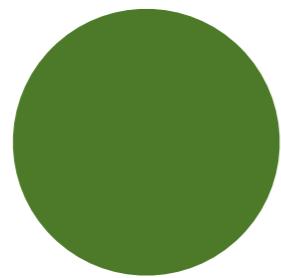
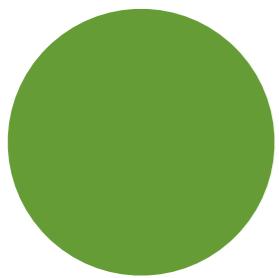
Why do we emphasize learning code?



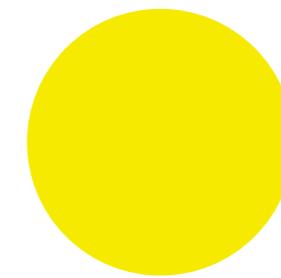
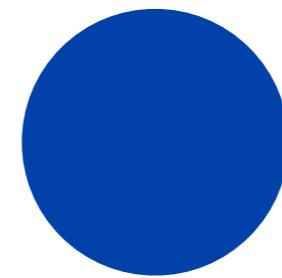


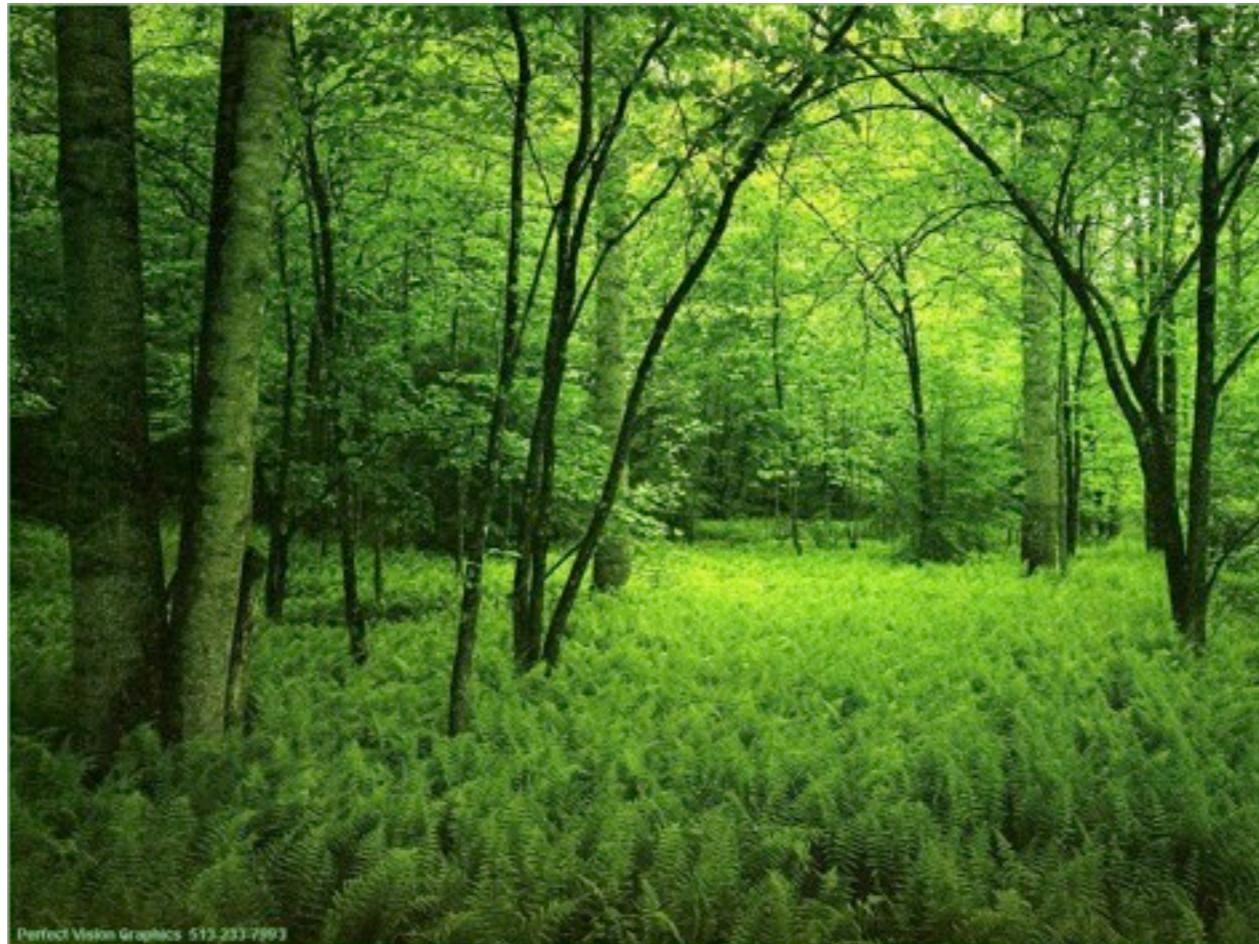


Colors you want

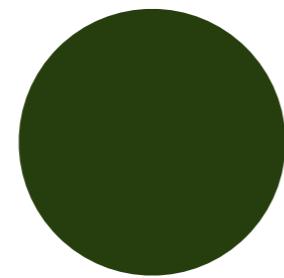
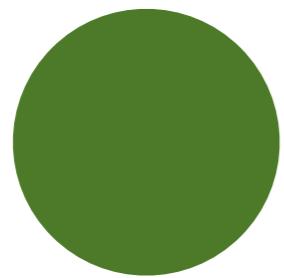
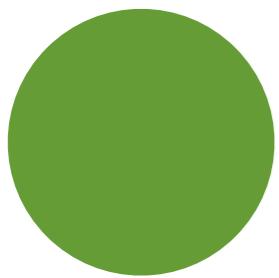


Colors the store has

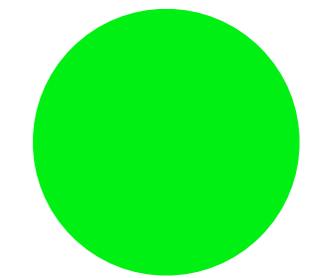
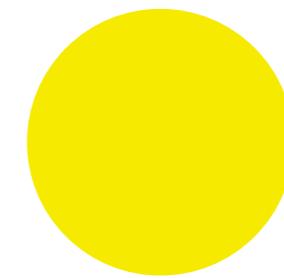
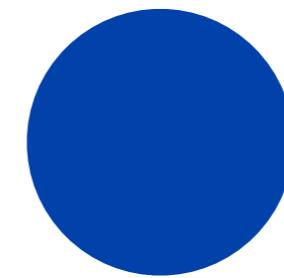




Colors you want

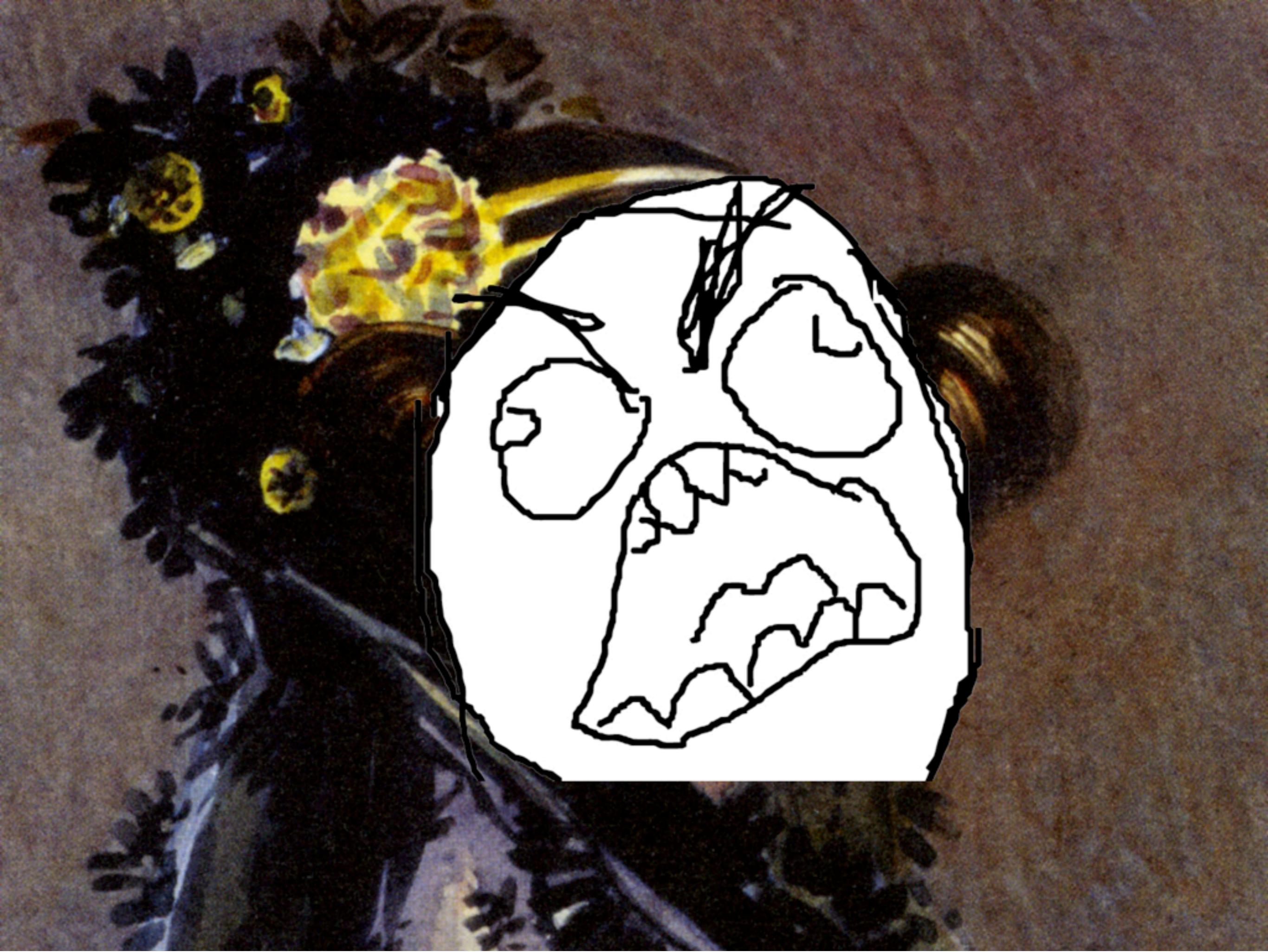


Colors the store has



A close-up painting of a woman's face. She has dark hair pulled back and is wearing a decorative headband with yellow and purple flowers. Her eyes are dark and expressive, looking slightly to the right. She has a neutral expression with a hint of a smile. The background is dark and textured.

What makes a good coder?



Myth: only bad programmers have to ask questions

[All Questions](#)
[newest](#)
[465 featured](#)
[frequent](#)
[votes](#)
[active](#)
[unanswered](#)
7015

votes

9
answers

369k views

[Why is processing a sorted array faster than an unsorted array?](#)

Here is a piece of C++ code that seems very peculiar. For some strange reason, sorting the data miraculously makes the code almost six times faster: #include <algorithm> #include <ctime> ...

[java](#) [c++](#) [performance](#) [optimization](#) [branch-prediction](#)

asked Jun 27 '12 at 13:51



GManNickG

141k • 18 • 235 • 384

4490

votes

22
answers

652k views

[How do I edit an incorrect commit message in Git?](#)

I stupidly did a Git commit while half asleep, and wrote the totally wrong thing in the commit message. How do I change the commit message? I have not yet pushed the commit to anyone.

[git](#) [version-control](#) [commit](#) [git-rewrite-history](#) [amend](#)

asked Oct 7 '08 at 15:44



Laurie Young

30.4k • 7 • 28 • 41

4073

votes

67
answers

570k views

[The Definitive C++ Book Guide and List](#)

This question attempts to collect the few pearls among the dozens of bad C++ books that are published every year. Unlike many other programming languages, which are often picked up on the go from ...

[c++](#) [c](#) [c++-faq](#)

community wiki

89 revs, 47 users 26%

sbi

3927

votes

22
answers

1.0m views

[How to undo the last Git commit?](#)

I accidentally added the wrong directory containing my files in Git. Instead of adding a .java file, I added the directory containing the .class file. How can I undo this action?

[git](#) [push](#) [undo](#) [git-commit](#)

community wiki

18 revs, 9 users 30%

Peter Mortensen

3902

votes

22
answers

1.0m views

[What is the correct JSON content type?](#)

I've been messing around with JSON for some time, just pushing it out as text and it hasn't hurt anybody (that I know of), but I'd like to start doing things properly. I have seen so many purported ...

asked Jan 25 '09 at 15:25

6,619,757

questions

 CAREERS 2.0

Programmer / Developer

Haverly Systems

Denville, NJ

Senior Ruby on Rails Developer

Blenderbox

New York, NY

C#/MVC/JQuery Developer For NY

FinTech Startup (Mid)

Advizr

New York, NY / remote

Market Risk/Counterparty Risk

Developer

Bloomberg

New York, NY

Senior Software Engineer- credit

products, \$15bn Hedge Fund

Pine River Capital Management

New York, NY

Software Developer

Two Sigma Investments

New York, NY / relocation

More jobs near Jersey City...

Related Tags

[c#](#) × 588809[java](#) × 568965[javascript](#) × 539671

We've found 675 code results

Sort: Best match ▾

[diraol/Amigos-da-Poli – frontpage-slider.js](#)

JavaScript

Last indexed 6 months ago

```
20 })(jQuery); //shoots self in head for doing this
21      //
22      // Kaloyan:
23      // I don't know why this works this way byt PhaseII use it this way in menu.js in pro;
```

[jajouka79/monstertuning-v1 – frontpage-slider.js](#)

JavaScript

Last indexed 6 months ago

```
20 })(jQuery); //shoots self in head for doing this
21      //
22      // Kaloyan:
23      // I don't know why this works this way byt PhaseII use it this way in menu.js in pro;
```

[aroxby-schell/Automatown – PrefabPlacer.cs](#)

C#

Last indexed 7 months ago

```
6     public GameObject prefab;
7     private LayerMask gridMask;
8
9     void Start()
10    {
11        //I don't know why this works, it seems I still don't understand Raycast
```

[jayferd/plans – production.rb](#)

Ruby

Last indexed 6 months ago

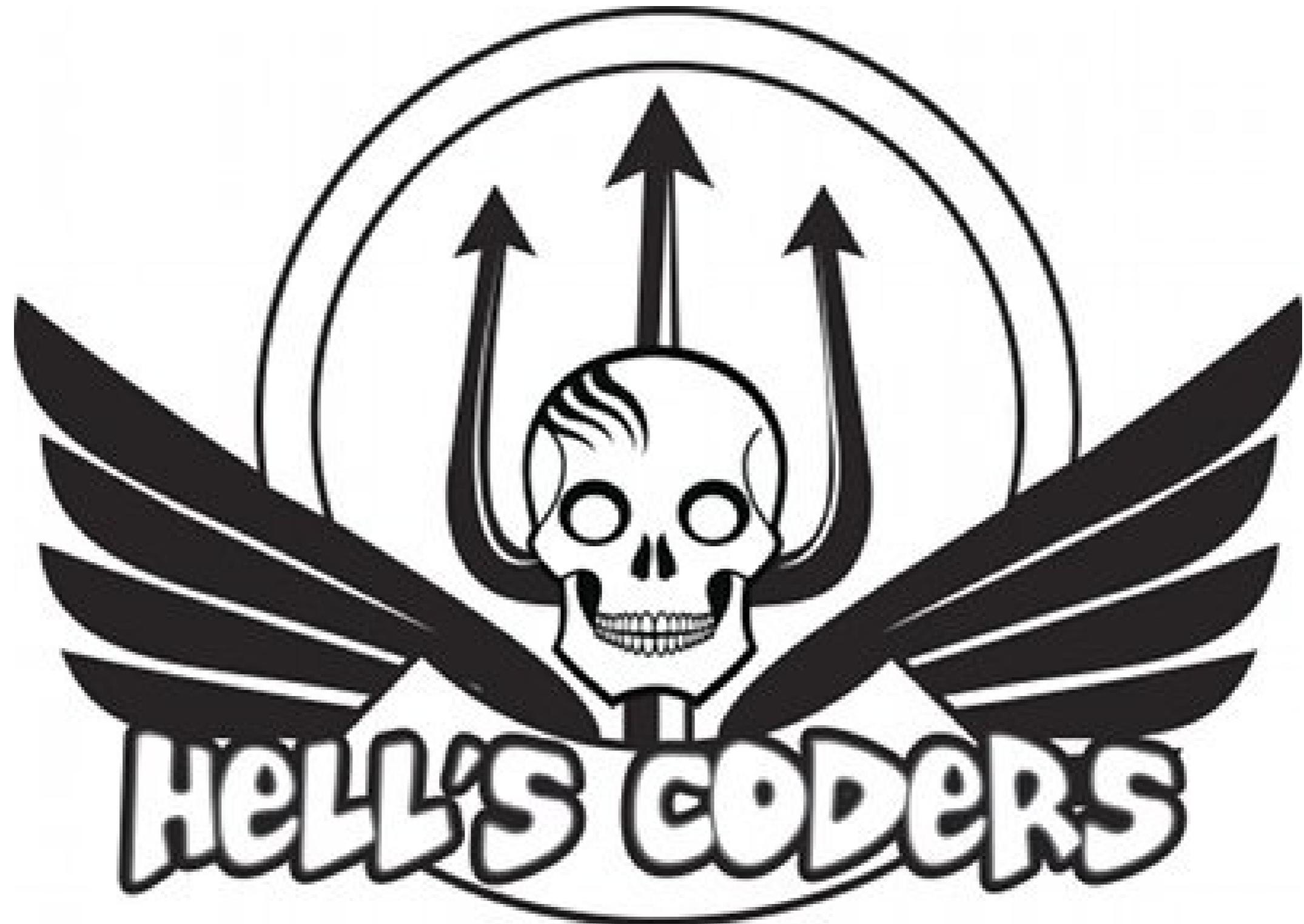
```
1 # hack to make the shared db work :(
```

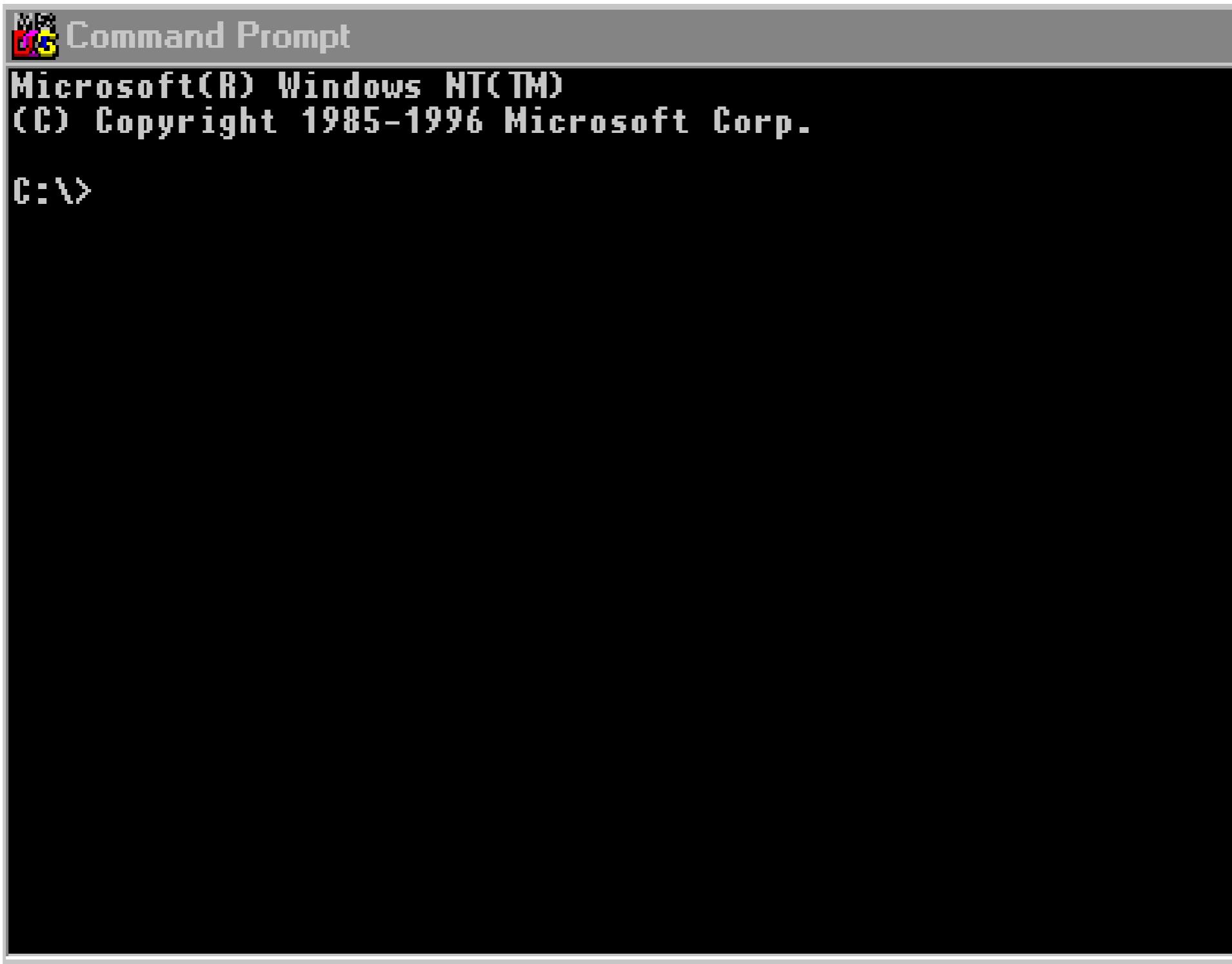


$$Y \times Y = 25$$

$$Y = 5, -5$$









HEY

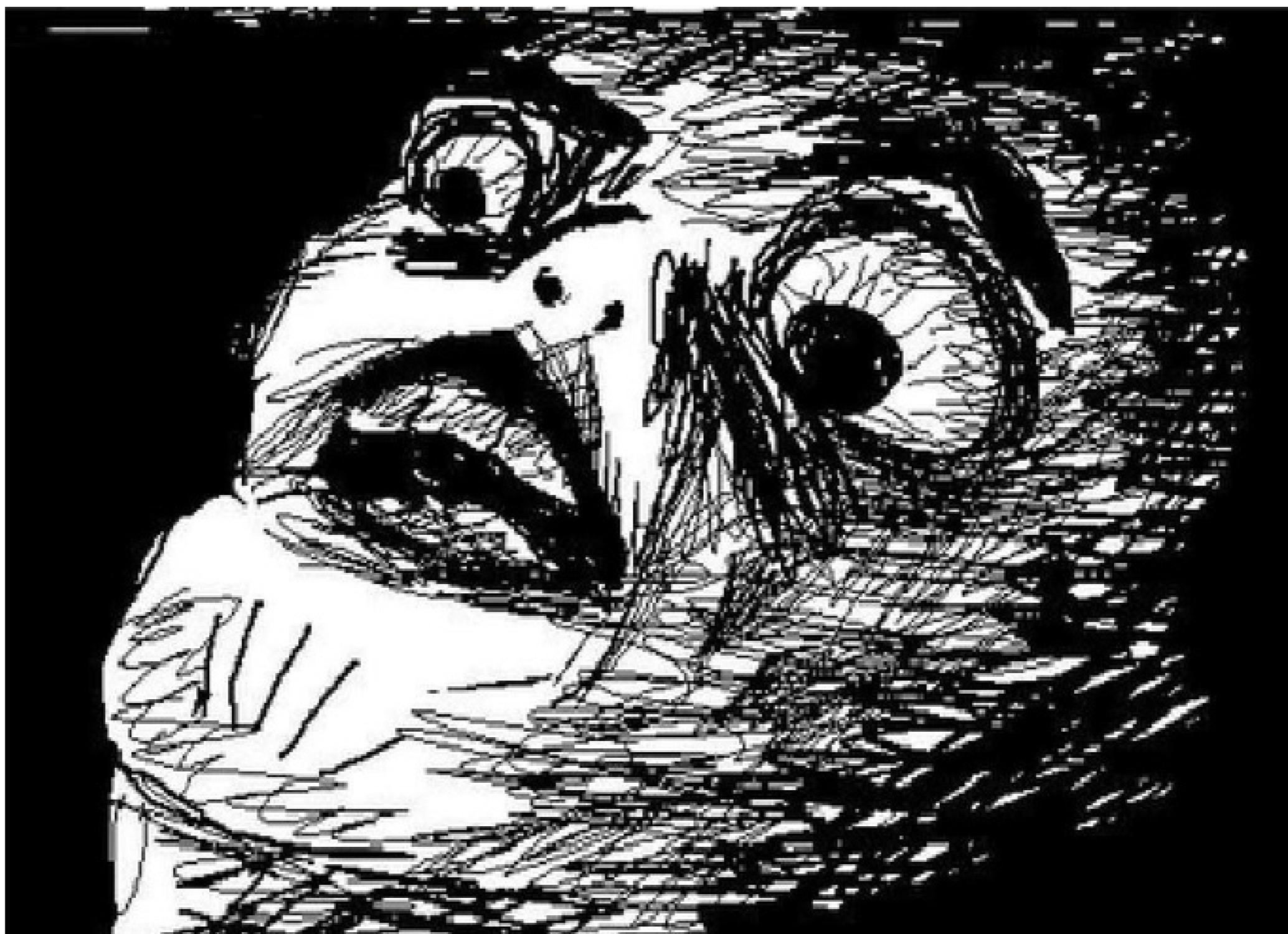
YOU CAN DO IT

Troll.me

**EVER TRIED.
EVER FAILED.
NO MATTER.
TRY AGAIN.
FAIL AGAIN.
FAIL BETTER.**

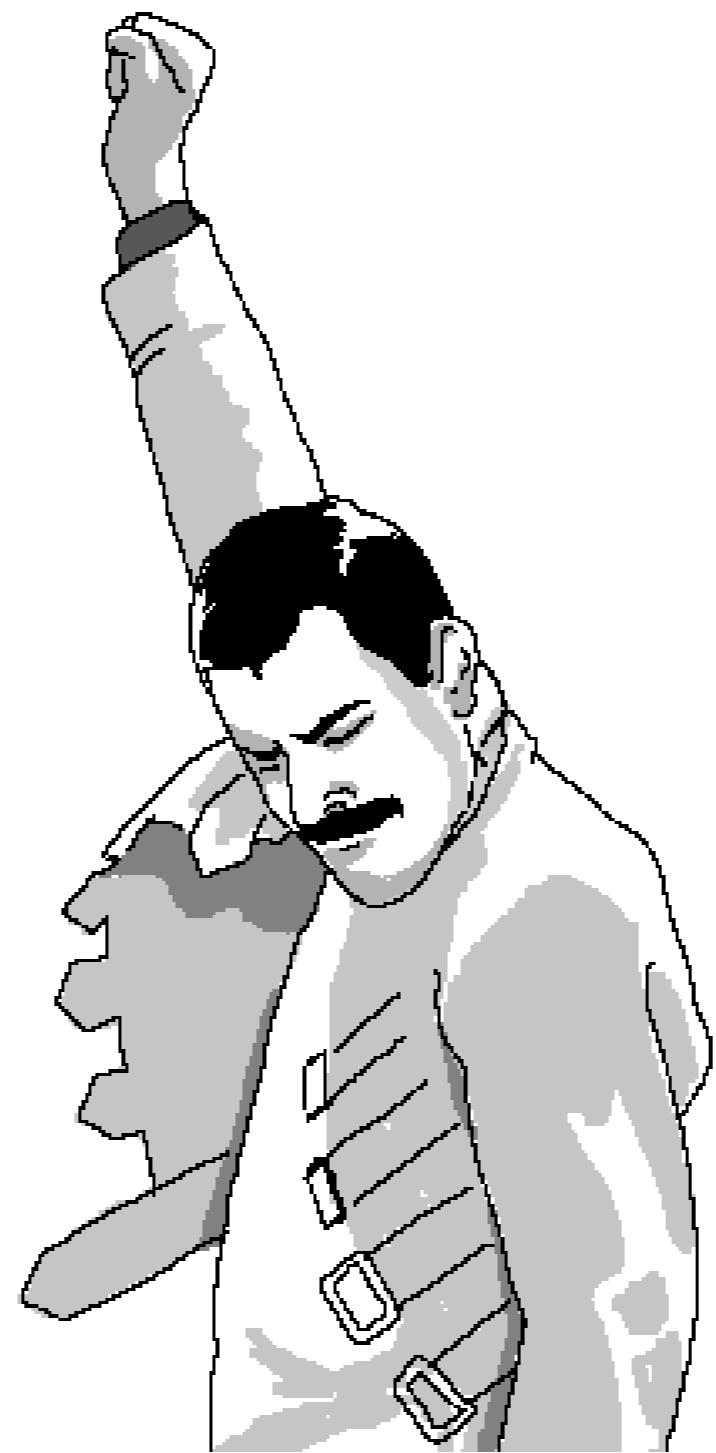
Samuel Beckett (1906-1989)



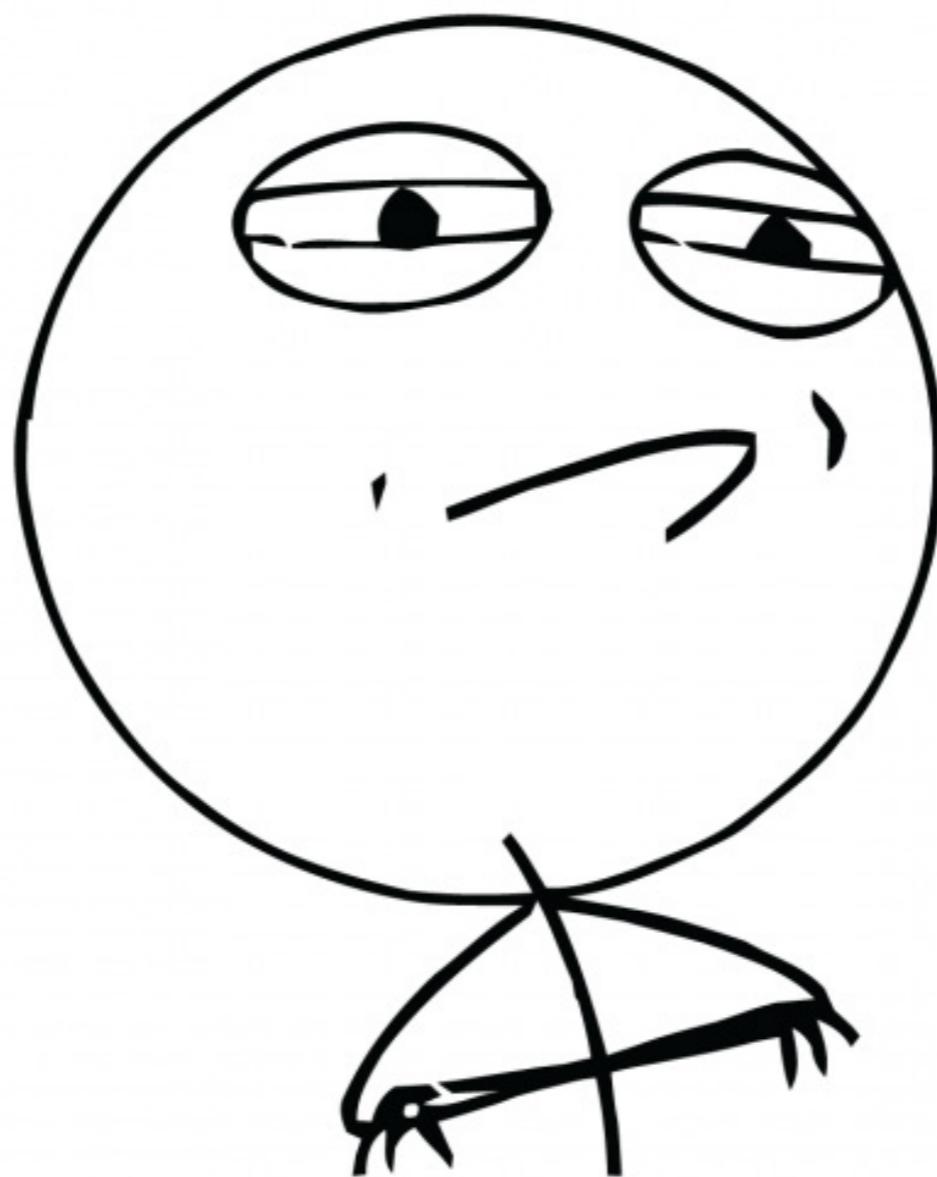




FFFFFFF
FFFFFFF
FFFFFFF
FFFUU
UUUU
UUUU
UUUU
UUUU
UUUU-



CHALLENGE ACCEPTED



[Cover](#)[Download](#)[Exhibition](#)[Reference](#)[Libraries](#)[Tools](#)[Environment](#)[Tutorials](#)[Examples](#)[Books](#)[» Download Processing](#)[» Play With Examples](#)[» Browse Tutorials](#)

Processing is a programming language, development environment, and online community. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. Initially created to serve as a software sketchbook and to teach computer programming fundamentals within a visual context, Processing evolved into a development tool for professionals. Today, there are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing

[» Exhibition](#)[Keyflies](#)

by Miles Peyton



What is Processing?

- » [Forum](#)
- » [GitHub](#)
- » [Issues](#)
- » [Wiki](#)
- » [FAQ](#)
- » [Twitter](#)
- » [Facebook](#)

[» For GNU/Linux, Mac OS X, and Windows](#)[» Over 100 libraries extend the core software](#)[» Well documented, with many books available](#)[» Hello Processing Videos](#)

This first look at Processing for total beginners is an introduction to programming in the context of the visual arts. Short video lessons introduce coding exercises that lead to designing an interactive drawing program. This experimental tutorial was created for Code.org's Hour of Code project for Computer Science Education Week.

[» Fragmented Memory](#)

by Phillip Stearns

[» Avena+ Test Bed](#)

by Benedikt Groß

[» T-shirts!](#)



What is Processing?

- Processing is a library for Java (a programming language)
 - Library: a collection of code intended to simplify coding process and/or give it more functionality
- We'll be working in the Processing environment (the app), which is also referred to as an IDE



So what's an IDE?

- IDE = Integrated Development Environment
- Basically just an application to help you write code in this language



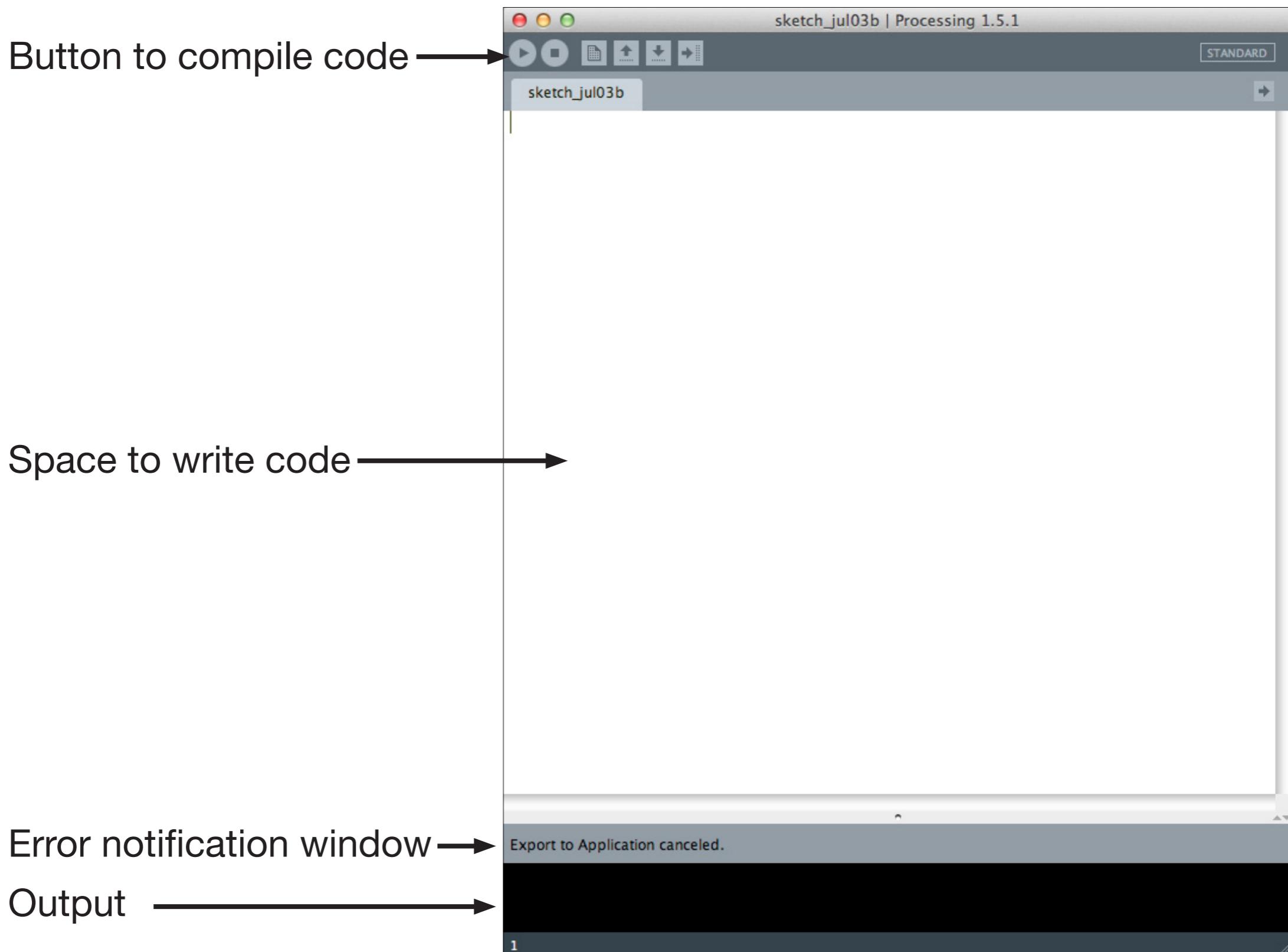
IDE: Cooking Simile

- Let's compare an IDE to the tools you need to create a delicious meal for royalty
- So what would you need?
 - A recipe (the instructions)
 - A stove/oven/microwave (something to take the raw materials and turn it into a meal)
 - A taste-tester (to make sure you don't accidentally poison anyone)



IDE: Cooking Simile

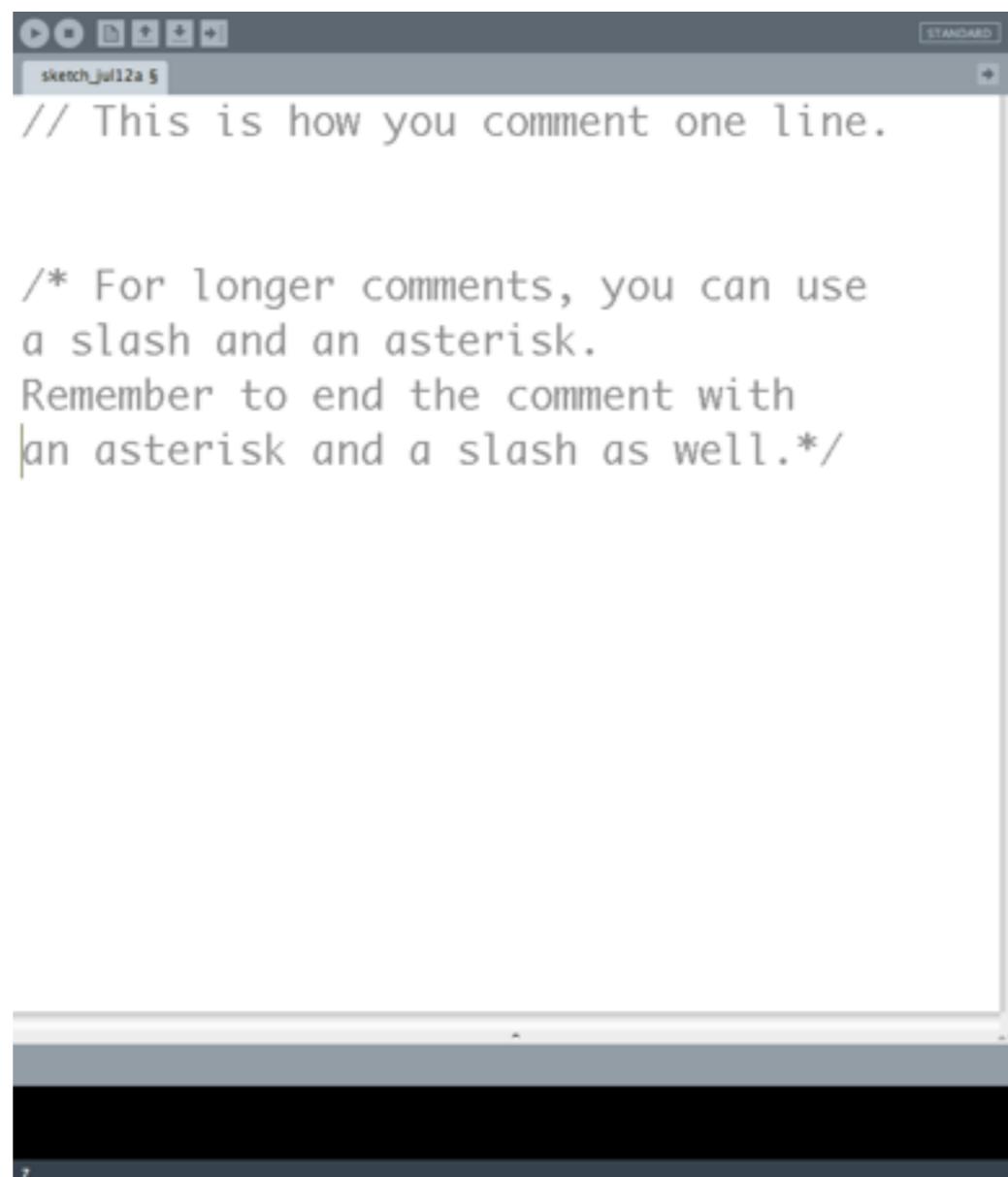
- Cooking:
- A recipe → • Written code
- A stove/oven → • A compiler
- A taste-tester → • A debugger



Taking Notes In Your Code

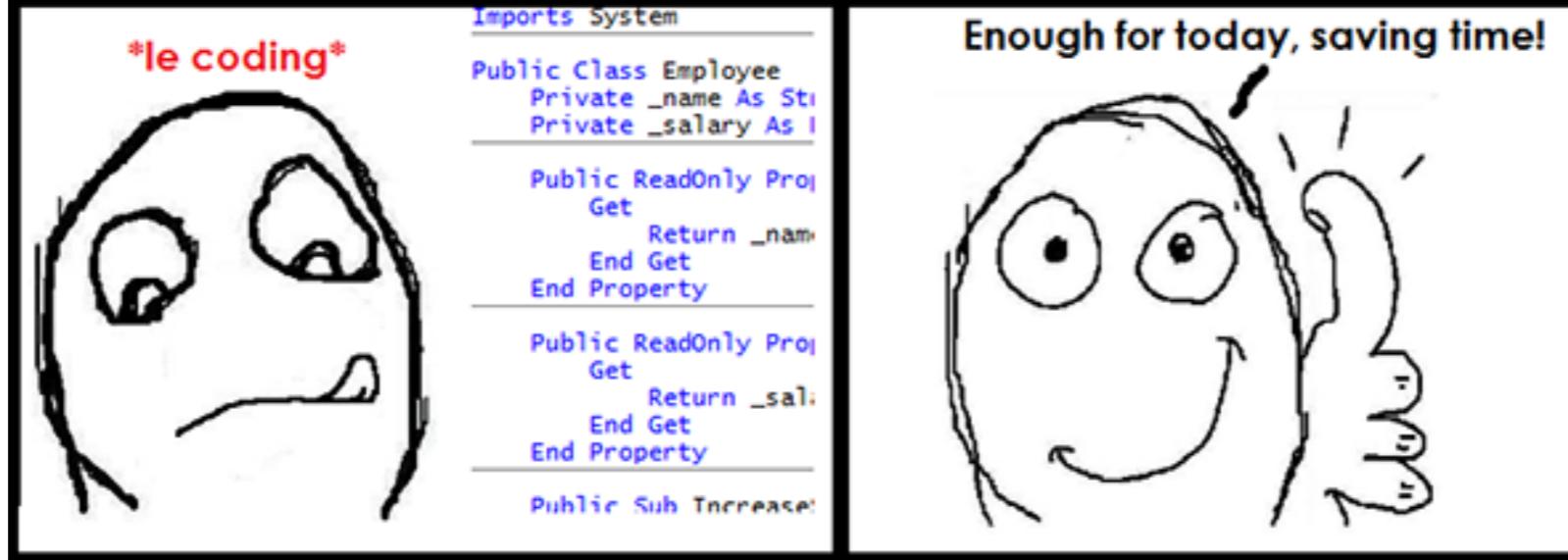


Commenting Code

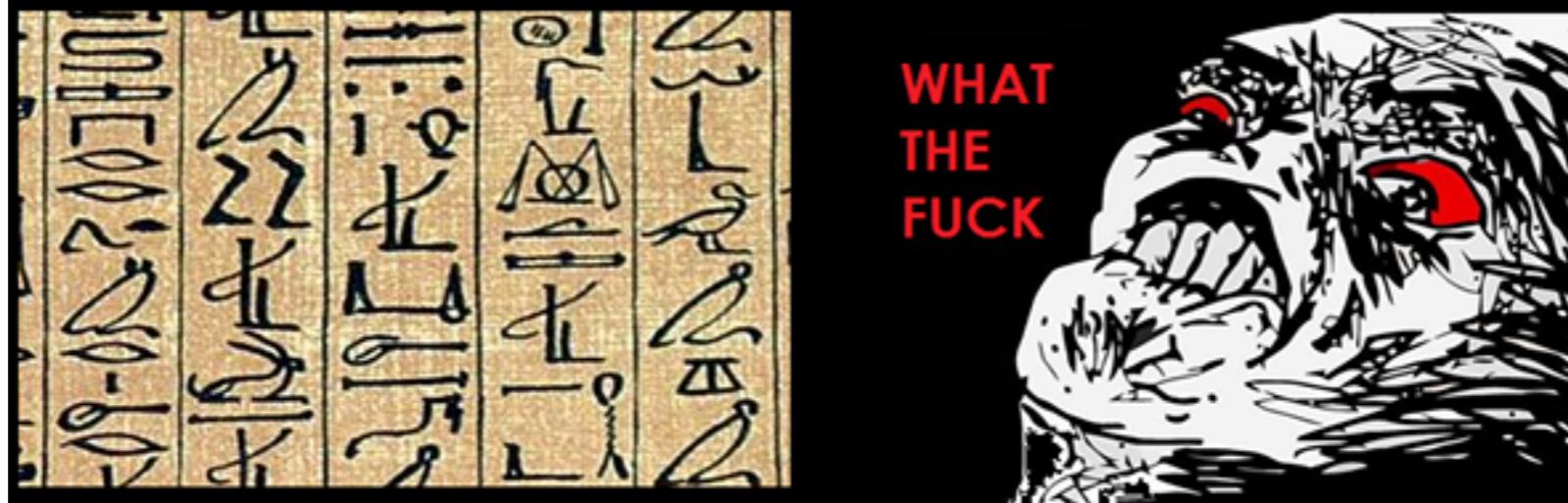


The screenshot shows the Arduino IDE interface with a sketch titled "sketch_jul12a". The code examples demonstrate two ways to comment code:

```
// This is how you comment one line.  
  
/* For longer comments, you can use  
a slash and an asterisk.  
Remember to end the comment with  
an asterisk and a slash as well.*/
```

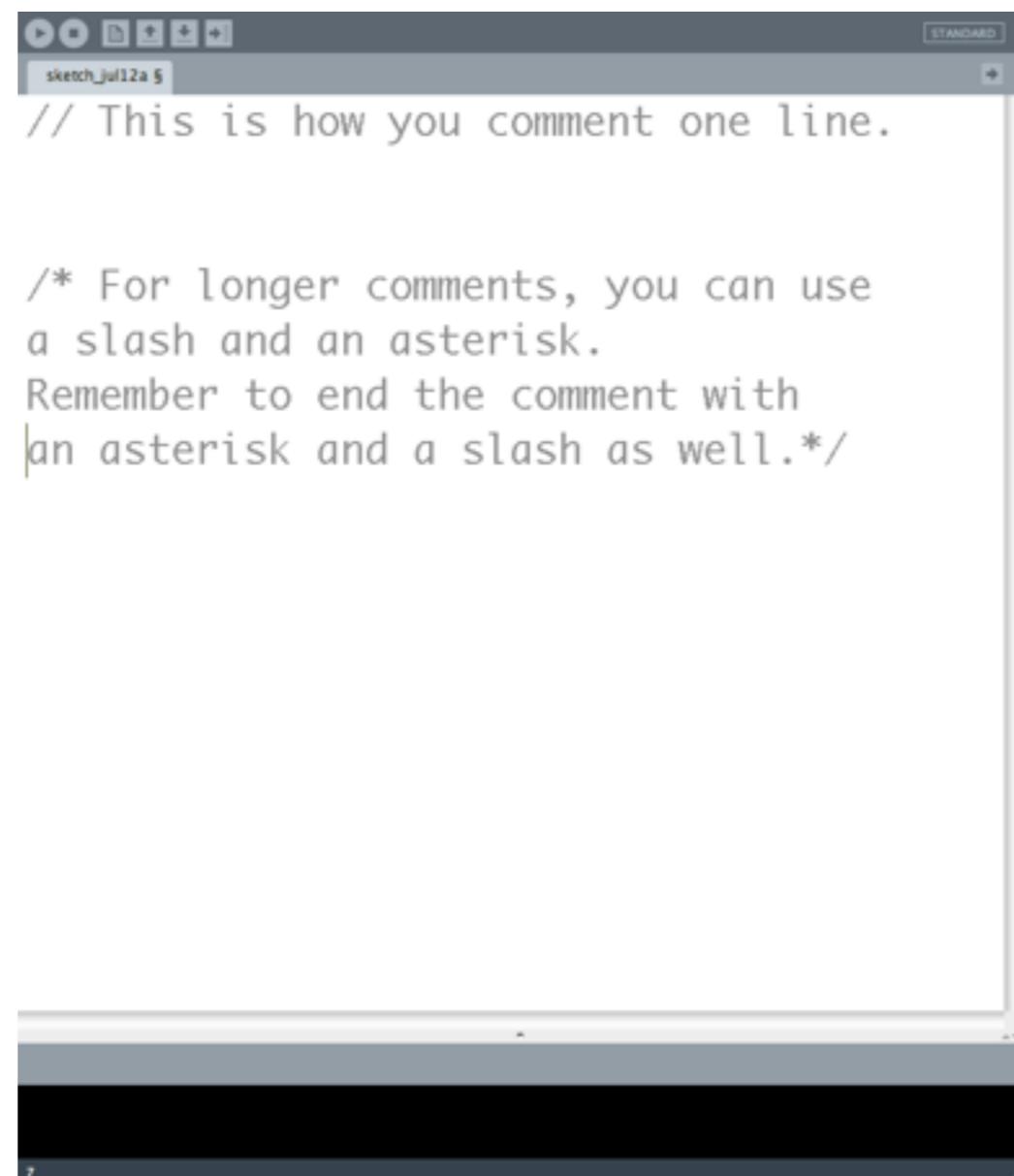


Opening file 6 weeks later...





Commenting Code, Again, 'Cause Seriously



```
// This is how you comment one line.

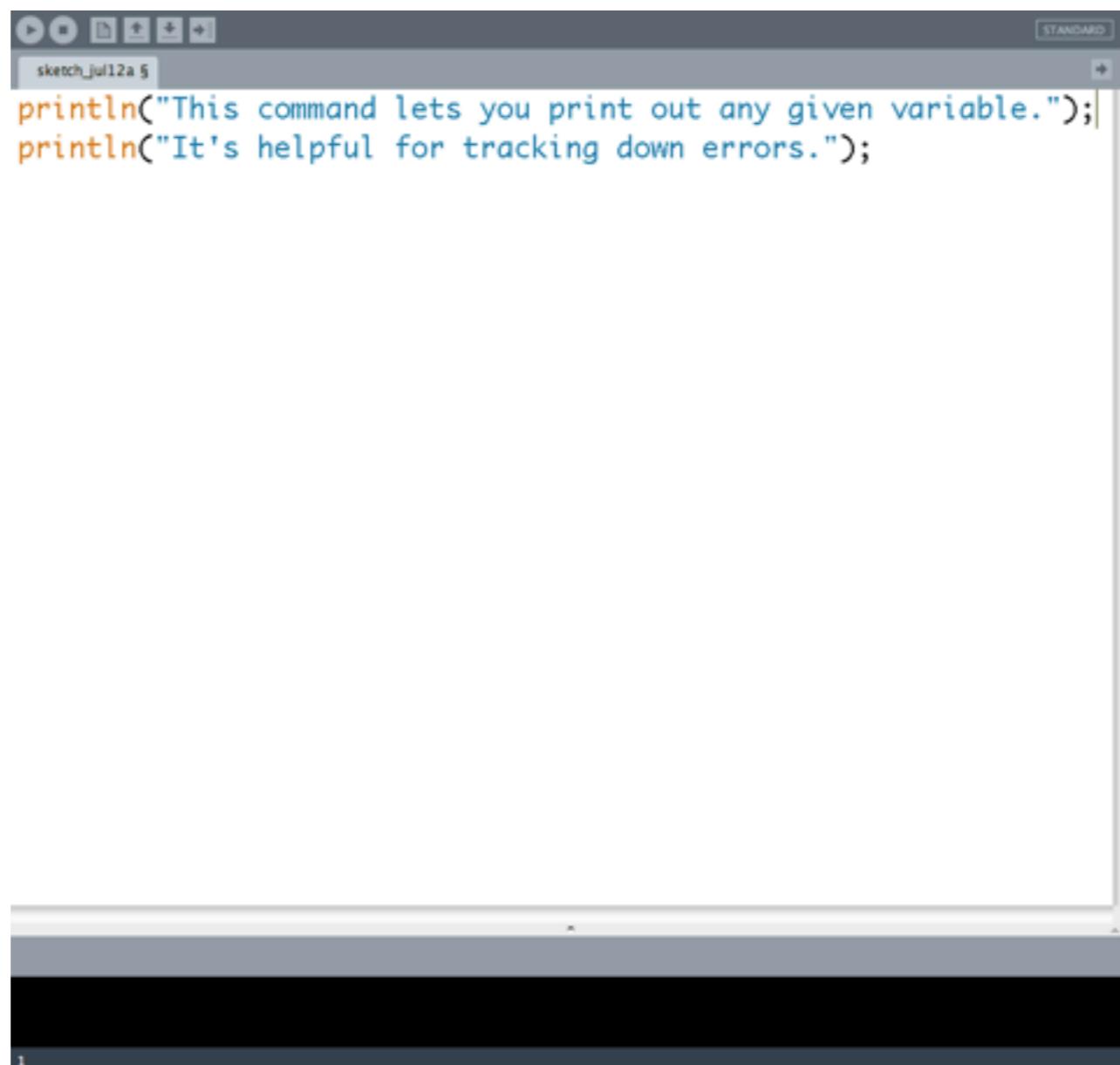
/* For longer comments, you can use
   a slash and an asterisk.
   Remember to end the comment with
   an asterisk and a slash as well.*/
```

A photograph of the front engine compartment of a dark-colored Chevrolet car. The hood is open, revealing the engine. The word "CHEVROLET" is printed in silver letters across the front edge of the hood. The engine features a prominent chrome air cleaner on top. Various hoses, belts, and mechanical components are visible. In the bottom left corner, a blue battery with "ACDelco" branding is partially visible.

Printing To Console



Printing to the Console

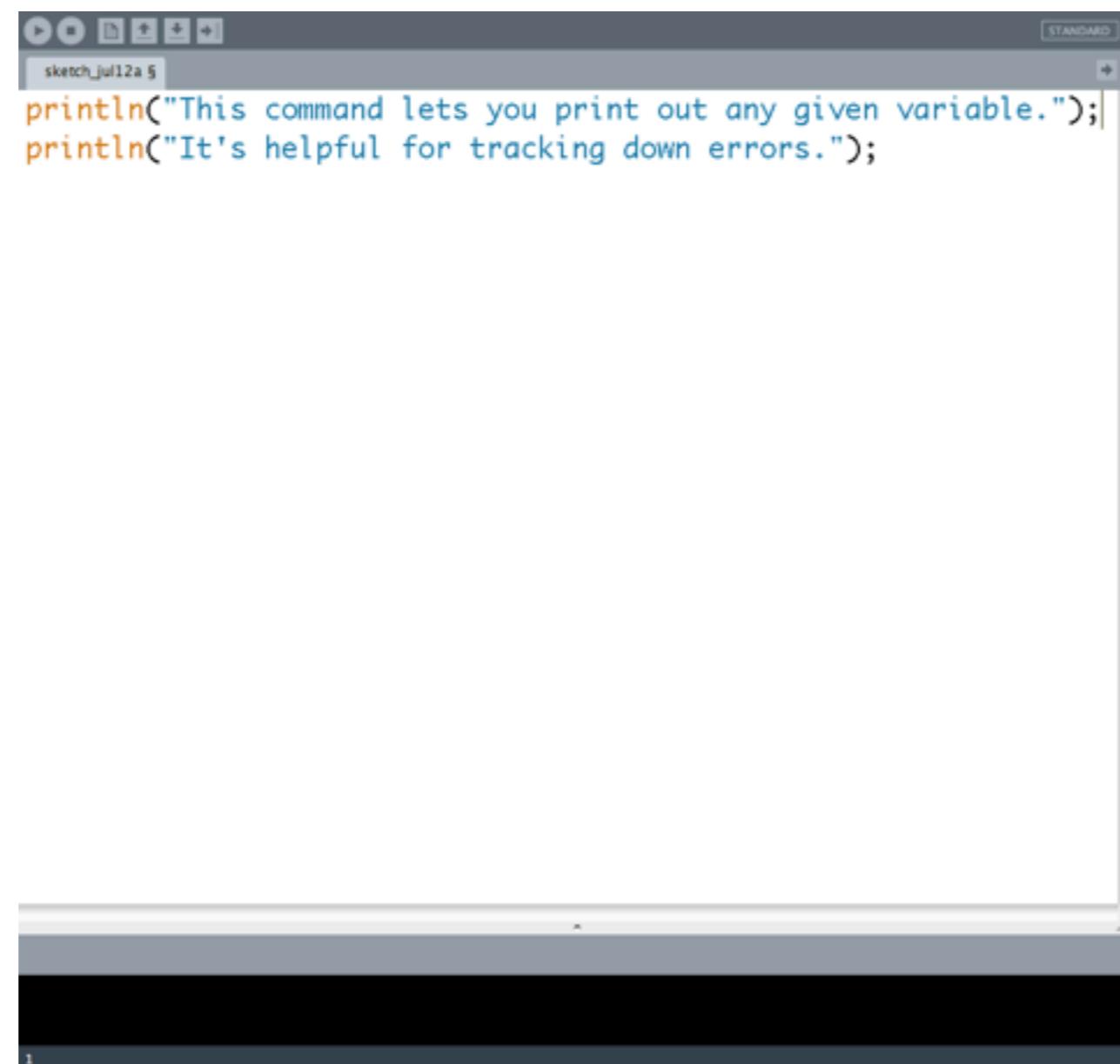


A screenshot of the Arduino IDE interface. The top menu bar shows 'STANDARD'. The code editor window contains the following code:

```
sketch_jul12a.ino
println("This command lets you print out any given variable.");
println("It's helpful for tracking down errors.");
```

The bottom status bar shows the number '1'.

Try using the `println()` command to print “Hello, world!”



The image shows a screenshot of the Arduino IDE. At the top, there's a toolbar with various icons. Below it is a menu bar with 'STANDARD' selected. The main window contains a code editor with a file named 'sketch_jul12a' (version 5). The code in the editor is:

```
println("This command lets you print out any given variable.");
println("It's helpful for tracking down errors.");
```



Answer

```
sketch_jul03c | Processing 1.5.1
STANDARD
sketch_jul03c §
println("Hello, world!");
```

Hello, world!

1

THE ELEMENTS OF STYLE

;

;

Syntax Proper
Proper Syntax

illustrated



Human Syntax

- End sentences with a punctuation mark
 - Hi there.
 - Where's the coffee?
- Use opening and closing punctuation
 - So I said, **"You look amazing!"**



Human Syntax

- Otherwise it would be very difficult to know where a sentence begins and you might say this is wrong but others could say maybe not it would also be difficult to know where it ends all of this is important for our comprehension of language and what it means otherwise we might misconstrue the meaning of a sentence you know people wouldn't want to read slides like this I say



Human Syntax

- Compare how grammar affects the following sentences:
 - “Jane said I looked nice.”
 - Jane said, “I looked nice.”



Human vs Code Syntax

Humans:

- End sentences with a punctuation mark
 - Hi there.
 - Where's the coffee?
- Use opening and closing punctuation
 - So I said, "You look amazing!"

Code:

- Generally end commands with a semicolon
 - `println("Syntax matters!");`
- Use opening and closing punctuation
 - `println("Syntax matters!");`



Parameters

- Parameters: information that affects how the computer executes a command

“Real Life”

```
rent_a_movie("Top Gun");
```

*Rent me a movie--
specifically, Top Gun.*

Code

```
println("Hello");
```

*Print to the console--
specifically, print Hello.*

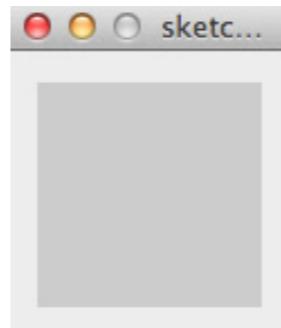


Let's Get Visual!





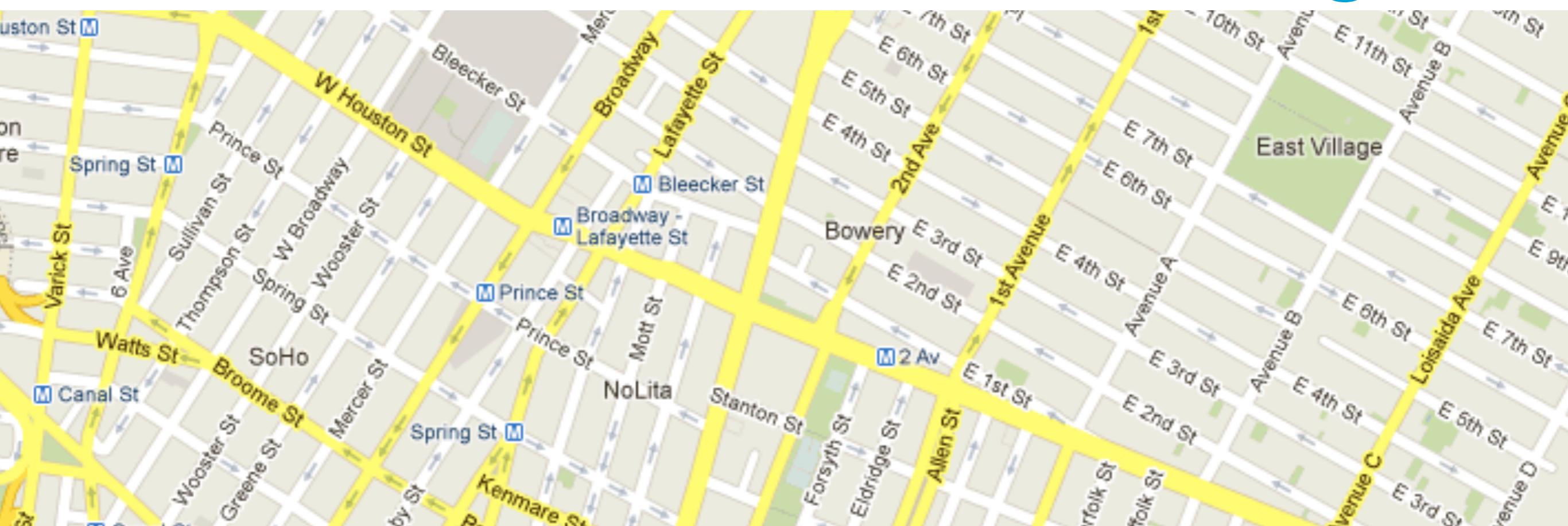
Basics of Visuals



- When you ran your sketch, you saw a little gray box pop up
- When you draw things, they show up in this window
- Default size: 100px by 100px (but we can change this!)



Locations in Processing





Locations In Processing

- In real life, having named and/or numbered streets and addresses helps us orient ourselves and figure out how to get where we're going
- Otherwise:
 - We might end up in the wrong place
 - Have no idea where to go, and just refuse to try

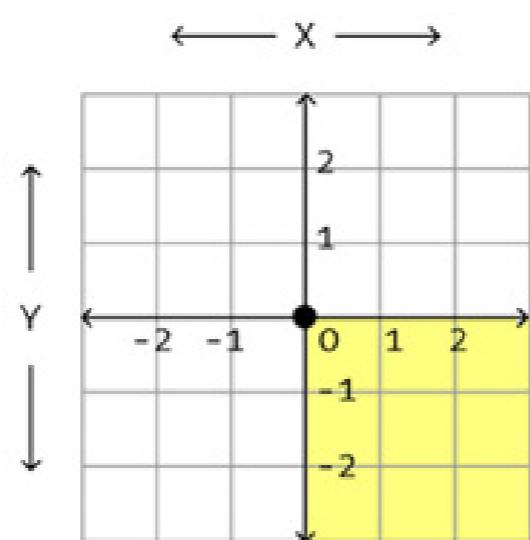


Coordinate Plane

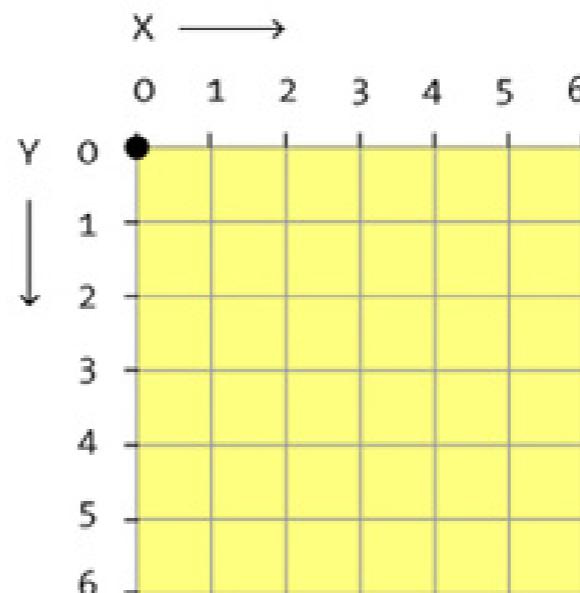
- Processing uses a grid system to know where to draw stuff to the screen
- Like a piece of graph paper underneath your sketch window



Coordinate Plane



Eighth Grade

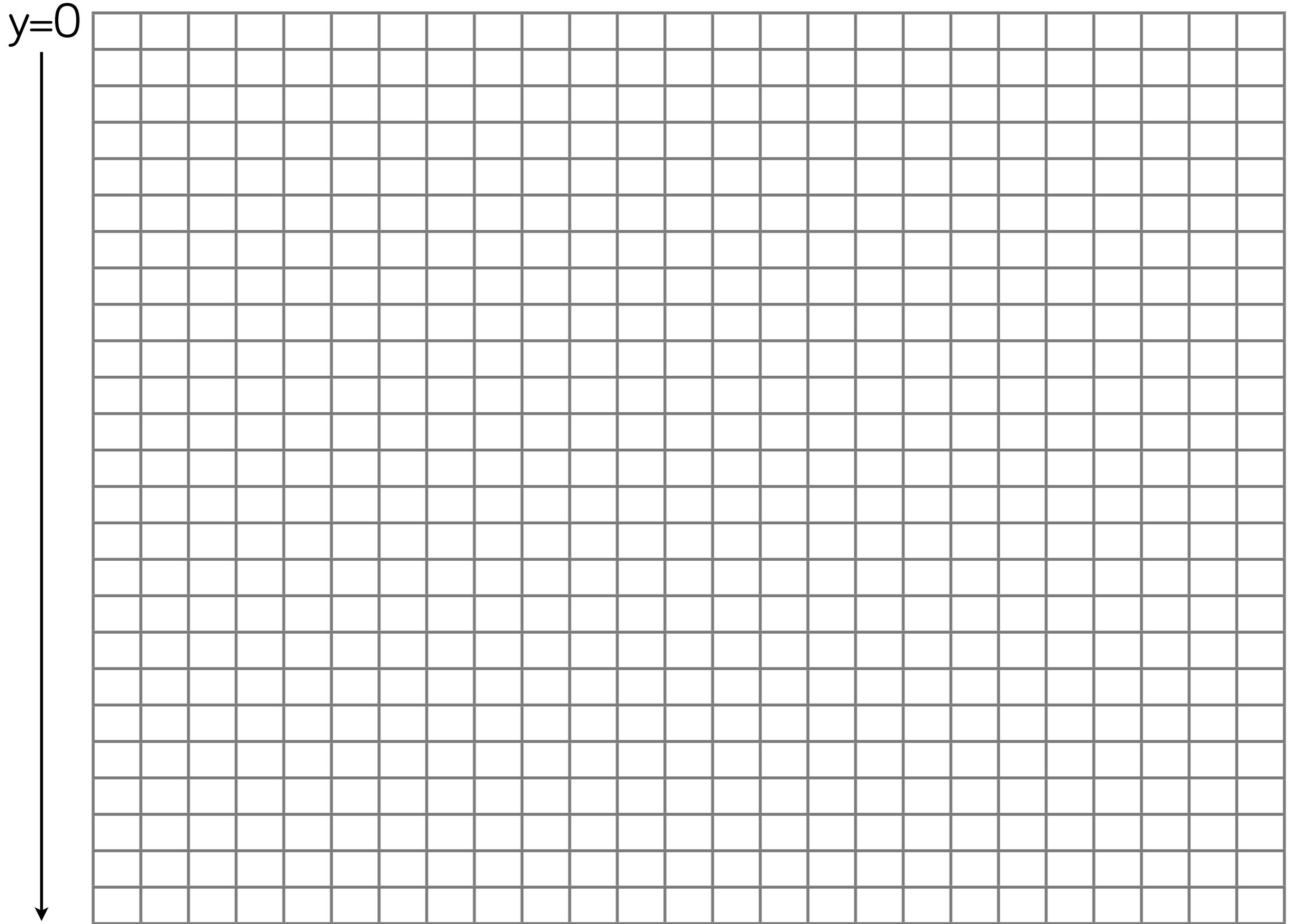


Computer

As you go **right**, the **x** value gets
bigger.

As you go **down**, the **y** value gets
bigger.

X=0 —————→ x=WIDTH



X=0 —————→ x=WIDTH



$y=HEIGHT$



Drawing Stuff



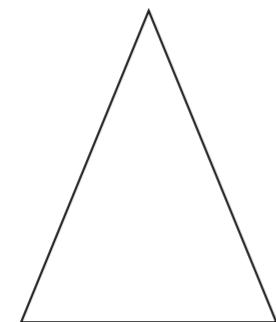
Drawing Stuff



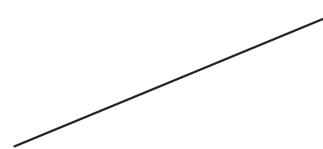
`ellipse(x position, y position, width, height);`



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



`triangle(x pos of 1st point, y pos of 1st point, x pos of 2nd point, y pos of 2nd point, x pos of 3rd point, y pos of 3rd point);`



`line(x pos of 1st point, y pos of 1st point, x pos of second point, y pos of 2nd point);`

.

`point(x pos, y pos);`



Drawing Stuff

Real Life

```
rent_a_movie("The Big Lebowski",  
TV_EDIT);
```

Rent me a movie--specifically, *The Big Lebowski*, *TV edited* version.

Coding

```
ellipse(10, 20, 50, 60);
```

Draw me an ellipse at *x-position* 10 and *y-position* 20, with a *width* of 50 pixels and a *height* of 60 pixels.



Parameters

- Different parameters = very different output!

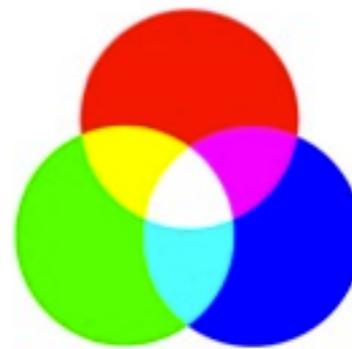




Coloring Stuff



Color



- RGB: Red, Green, Blue
- Additive color: the higher the values we give these three colors, the closer we get to white



Color

- R, G, and B can have values from 0 (least) to 255 (most)



Red: 255
Green: 242
Blue: 0



Red: 37
Green: 64
Blue: 143



Red: 37
Green: 216
Blue: 48

Red: 255
Green: 255
Blue: 255



Fill and Stroke



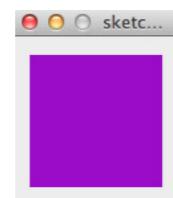
```
fill(210, 35, 42);  
ellipse(x position, y position, width, height);
```



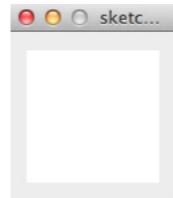
```
stroke(210, 35, 42);  
ellipse(x position, y position, width, height);
```



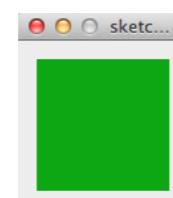
```
noStroke();  
fill(210, 35, 42);  
ellipse(x position, y position, width, height);
```



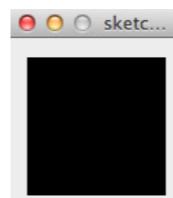
```
background(155, 12, 200);
```



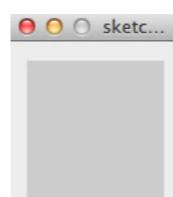
```
background(255);
```



```
background(13, 167, 19);
```



```
background(0);
```



```
background(100);
```

A close-up photograph of a pile of colorful crayons, showing various colors like pink, yellow, green, blue, and purple. Some crayons have their brand names visible on the labels, such as "CRA-Z-ART" and "CRAYOLA".

Coloring Stuff



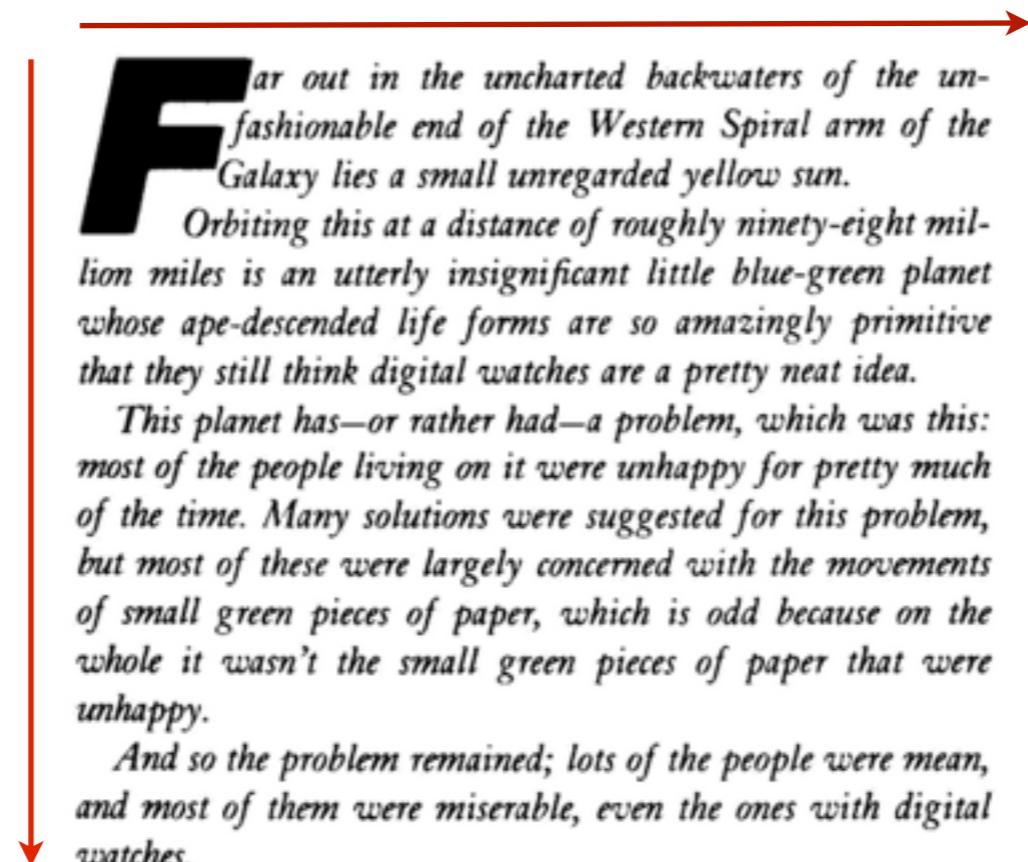
Inherent Order

Far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun.

Orbiting this at a distance of roughly ninety-eight million miles is an utterly insignificant little blue-green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea.

This planet has—or rather had—a problem, which was this: most of the people living on it were unhappy for pretty much of the time. Many solutions were suggested for this problem, but most of these were largely concerned with the movements of small green pieces of paper, which is odd because on the whole it wasn't the small green pieces of paper that were unhappy.

And so the problem remained; lots of the people were mean, and most of them were miserable, even the ones with digital watches.





Inherent Order

Humans

Far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun.

Orbiting this at a distance of roughly ninety-eight million miles is an utterly insignificant little blue-green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea.

This planet has—or rather had—a problem, which was this: most of the people living on it were unhappy for pretty much of the time. Many solutions were suggested for this problem, but most of these were largely concerned with the movements of small green pieces of paper, which is odd because on the whole it wasn't the small green pieces of paper that were unhappy.

And so the problem remained; lots of the people were mean, and most of them were miserable, even the ones with digital watches.

Processing

The screenshot shows the Processing IDE interface. The code window contains the following Java-like pseudocode:

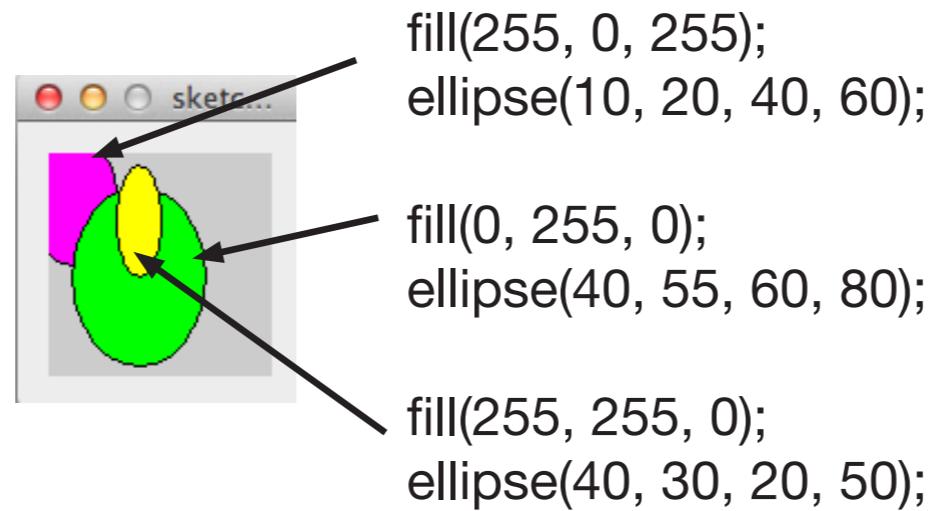
```
String name = "Jane";
name = "Rebecca";
println(name);
```

The output window below shows the result of the code execution: "Rebecca".

We'll talk about automatically repeating these commands later.



Inherent Order



- Functions execute from top to bottom
- If you use a drawing command higher up, it will be drawn below any later drawn stuff

Try drawing some shapes! See how the order in which you write your commands affects how they're drawn.

Variables and Datatypes



Why do we need variables?

- In this sketch, we draw a bunch of shapes
- They each have an x-value of 20
- What if we wanted to move all the shapes over by 5?

```
sketch_140209a | Processing 2.0b8
sketch_140209a §
ellipse(20, 15, 40, 40);
ellipse(20, 18, 10, 10);
ellipse(20, 23, 30, 30);
```



Why do we need variables?

- If we want to change the x-value, we'd have to change it *three times*
- It would be a lot easier if we only had to change it once

A screenshot of the Processing 2.0b8 software interface. The title bar says "sketch_140209a | Processing 2.0b8". The toolbar includes icons for play, stop, file, and other sketch controls. The code editor shows the following Java code:

```
sketch_140209a §
ellipse(20, 15, 40, 40);
ellipse(20, 18, 10, 10);
ellipse(20, 23, 30, 30);
```

The preview window below shows three ellipses at different positions and sizes.



Why do we need variables?

- This is where variables come in!
- **Variable**: a container for a piece of information that you can reference throughout your code
- Especially useful for data you might want to change

sketch_140209a | Processing 2.0b8

sketch_140209a §

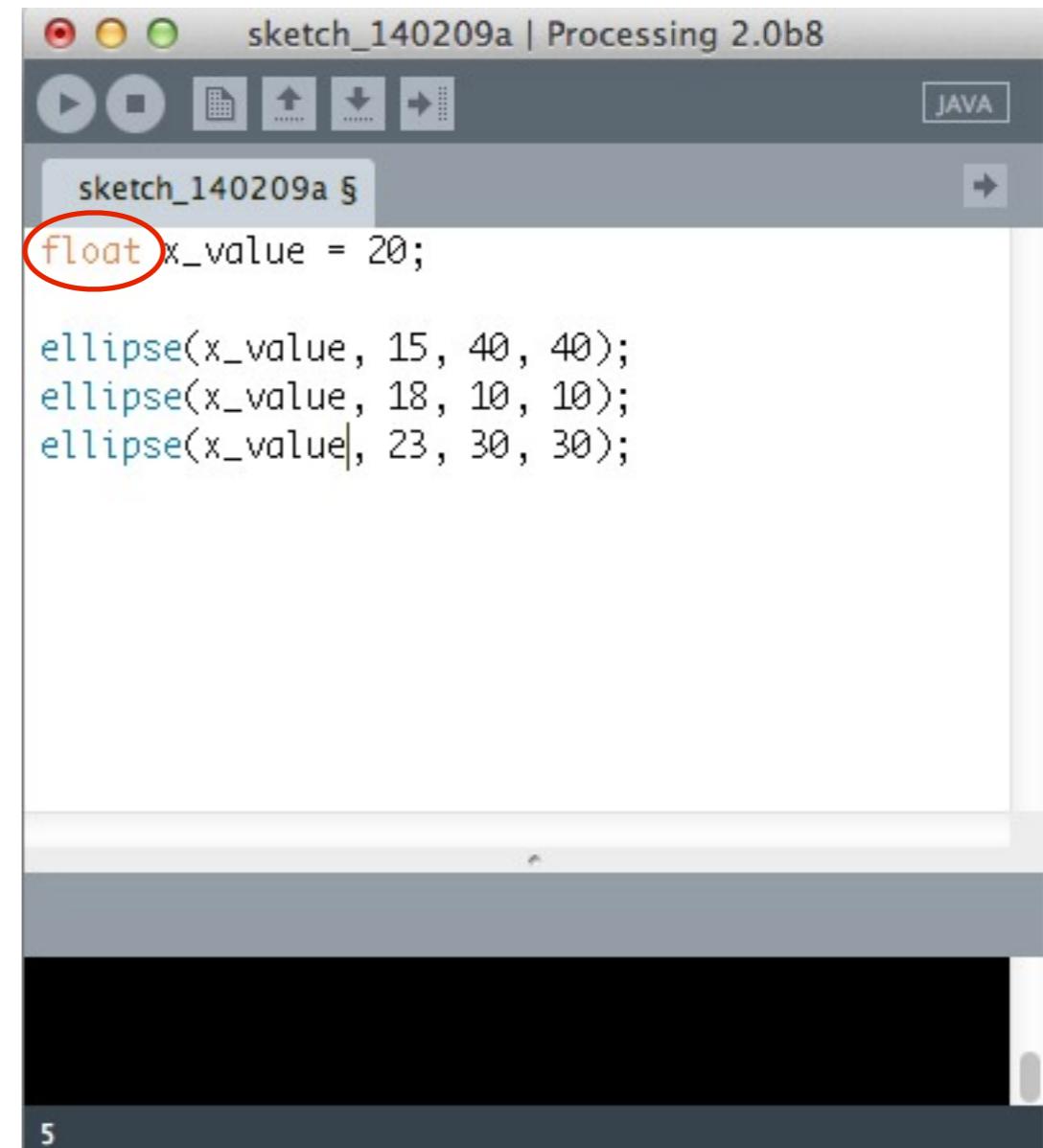
```
float x_value = 20;  
ellipse(x_value, 15, 40, 40);  
ellipse(x_value, 18, 10, 10);  
ellipse(x_value, 23, 30, 30);
```

5



Why do we need variables?

- Notice the word in orange before the variable name? That's its **datatype**



```
sketch_140209a | Processing 2.0b8
sketch_140209a §
float x_value = 20;
ellipse(x_value, 15, 40, 40);
ellipse(x_value, 18, 10, 10);
ellipse(x_value, 23, 30, 30);
```



Datatypes

- We don't think about it much IRL, but we need some context in order to store and process the information we're given



Datatypes

Name	Use	Code	Real life
<i>Integer</i>	Whole numbers	int	I have 1 dog.
<i>Float</i>	Numbers with a decimal	float	I'm 25.1 years old.
<i>String</i>	Words or letters	String	"Get off my lawn!"
<i>Character</i>	A single letter	char	'A'
<i>Boolean</i>	True or false	boolean	It's true that I want coffee.

What datatype would you use to describe these things?

Choose: integer, float, string, boolean, char

- The number of kids you have
- Your name
- Your middle initial
- Your apartment number
- Whether you ate breakfast or not
- The street you live on
- The temperature outside
- Whether the lights are on or not



Summary

- Variables are used to store pieces of information that might change, and/or that are used multiple times
- When you make a variable, you have to tell the computer what kind of data that variable will hold



Declaring and Initializing Variables

Datatype

What kind of thing are we dealing with here? Can we add it or subtract it? Is it text? Or is it a true or false condition?

Name

What are we calling this thing? We need a way to refer to it within our code.

Value

What actual information is this variable representing?

Syntax: datatype name = value;



Variables and Datatypes

- Let's say I'm making a visualization of how many hours of code Code Liberation has taught
- Since we're always teaching code, that number could change--so I'd want to make a variable to store it

```
int hours_of_code_taught = 143;
```

datatype variable name value



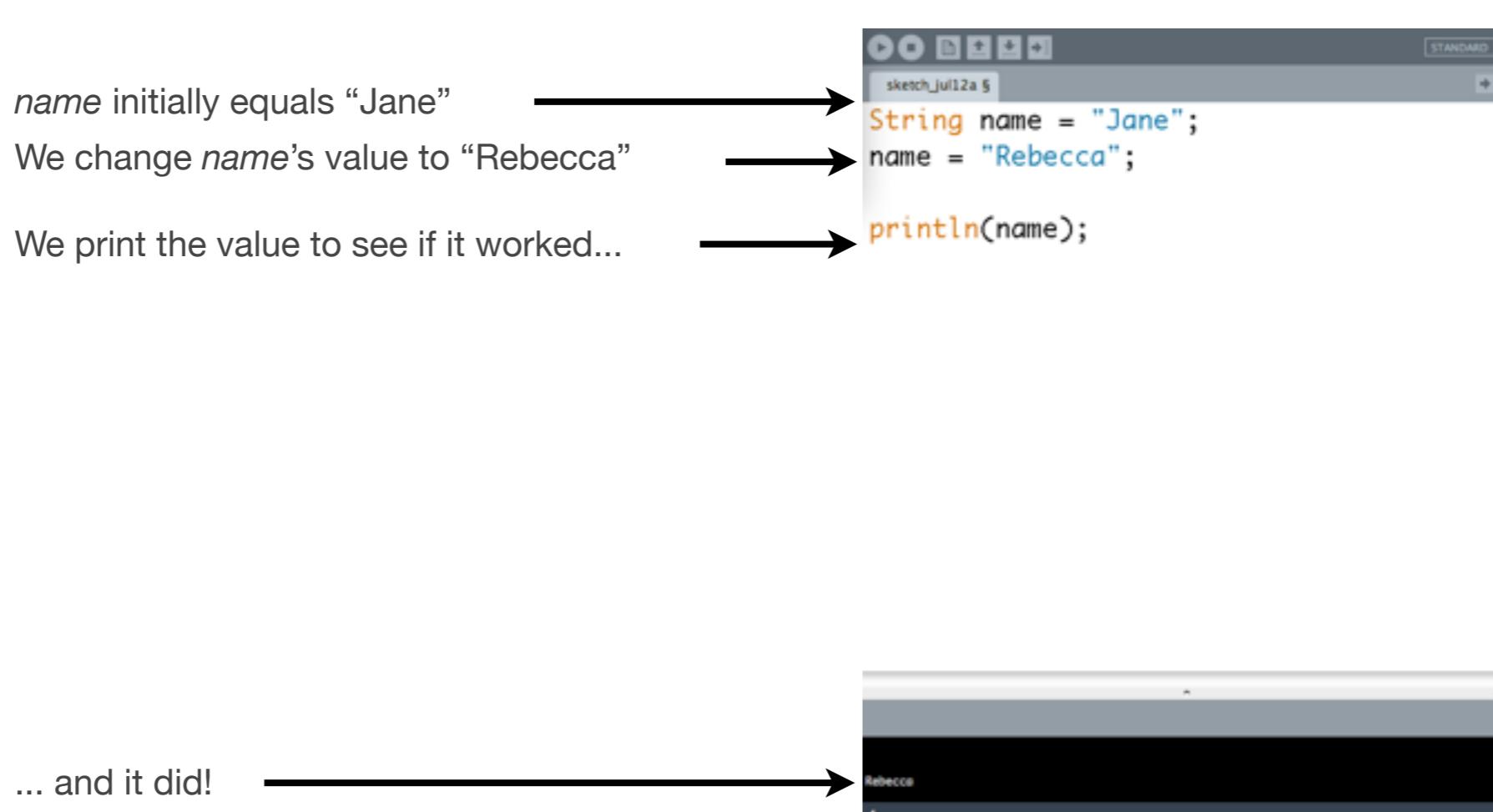
Naming Conventions

- Use underscores or alternating caps (camel case) to make things readable
 - Good: user_age, player_speed
 - Bad: timeelapsesincethegamestarted
- Name your variable something that makes sense
 - Good: player_health
 - Bad: data, aksdljfl, my_var
- Variables cannot start with a number, but can contain numbers
 - Good: player_1_health
 - Bad: 1_player_health



Changing Variable Values

Use a single-equals (=) to make the thing on the *left* equal the thing on the *right*.





Mathematical Operators

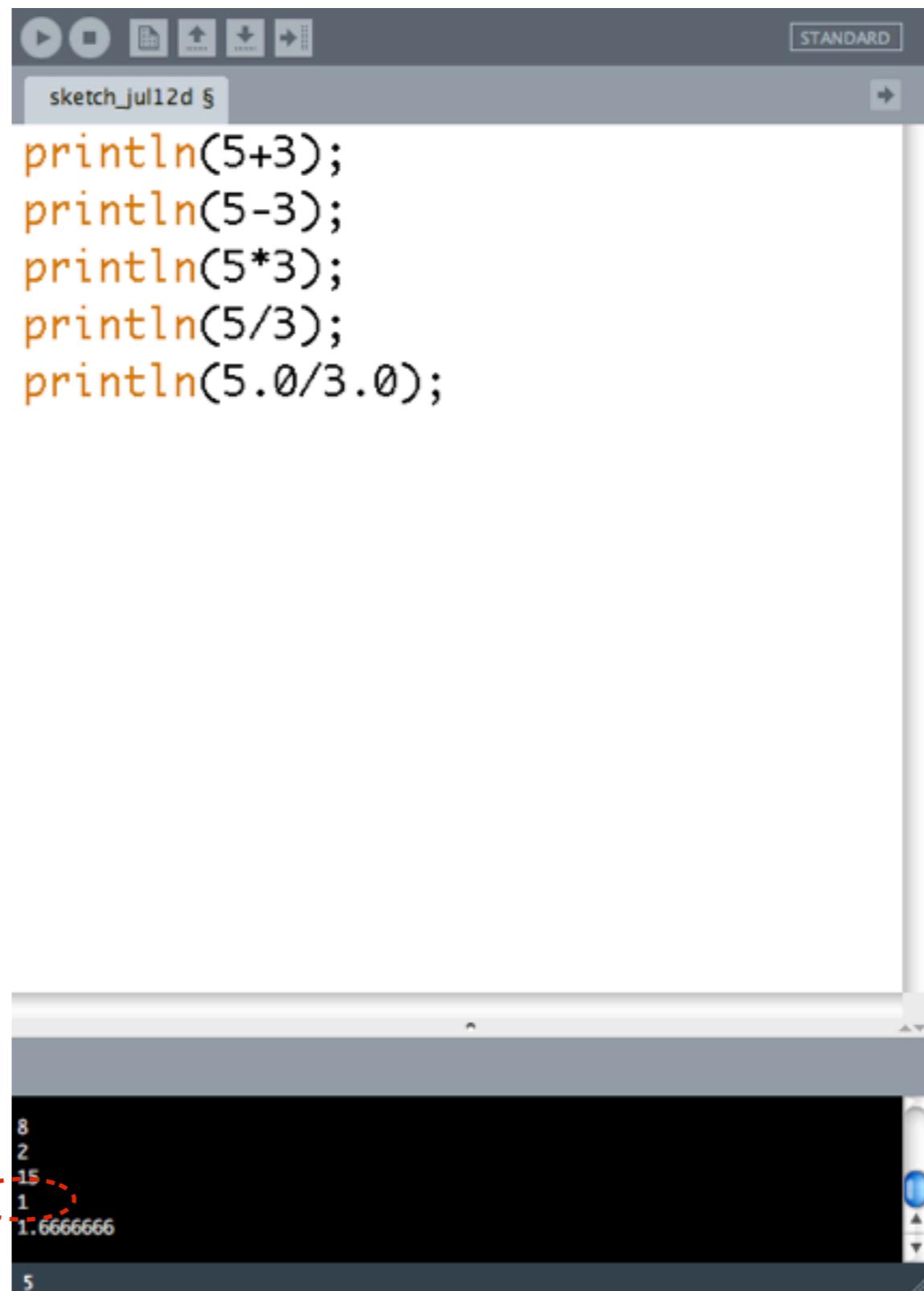
Operator	Code
Addition	+
Subtraction	-
Division	/
Multiplication	*

We'll talk about another operator, modulo (%), soon.

Try using `println` and the math symbols to print the result of the following equations.

- Five plus 3
- Five minus 3
- Five times 3
- Five divided by three
- Five-point-zero divided by three-point-zero

- Notice something weird?
- To a computer, $5/3$ is not the same as $5.0/3.0$!
- It goes back to datatypes--the computer sees whole numbers, so it spits out a whole number--even when that's not really correct



```
sketch_jul12d §
println(5+3);
println(5-3);
println(5*3);
println(5/3);
println(5.0/3.0);
```

The screenshot shows the Arduino IDE interface. The top bar has standard icons and the text "STANDARD". Below is the code editor with the following content:

```
sketch_jul12d §
println(5+3);
println(5-3);
println(5*3);
println(5/3);
println(5.0/3.0);
```

The bottom half of the screen shows the串行 monitor (Serial Monitor) with the following output:

```
8
2
15
1
1.6666666
```

The number "1.6666666" is circled in red.

Try defining an integer variable called `xPos`, and make it equal 5.

Define another integer variable called `yPos`, and make it equal 7.

Draw a circle at `xPos` and `yPos`.

Draw another circle at `xPos + 10` and `yPos`.

Draw another circle at `xPos + 40` and `yPos`.

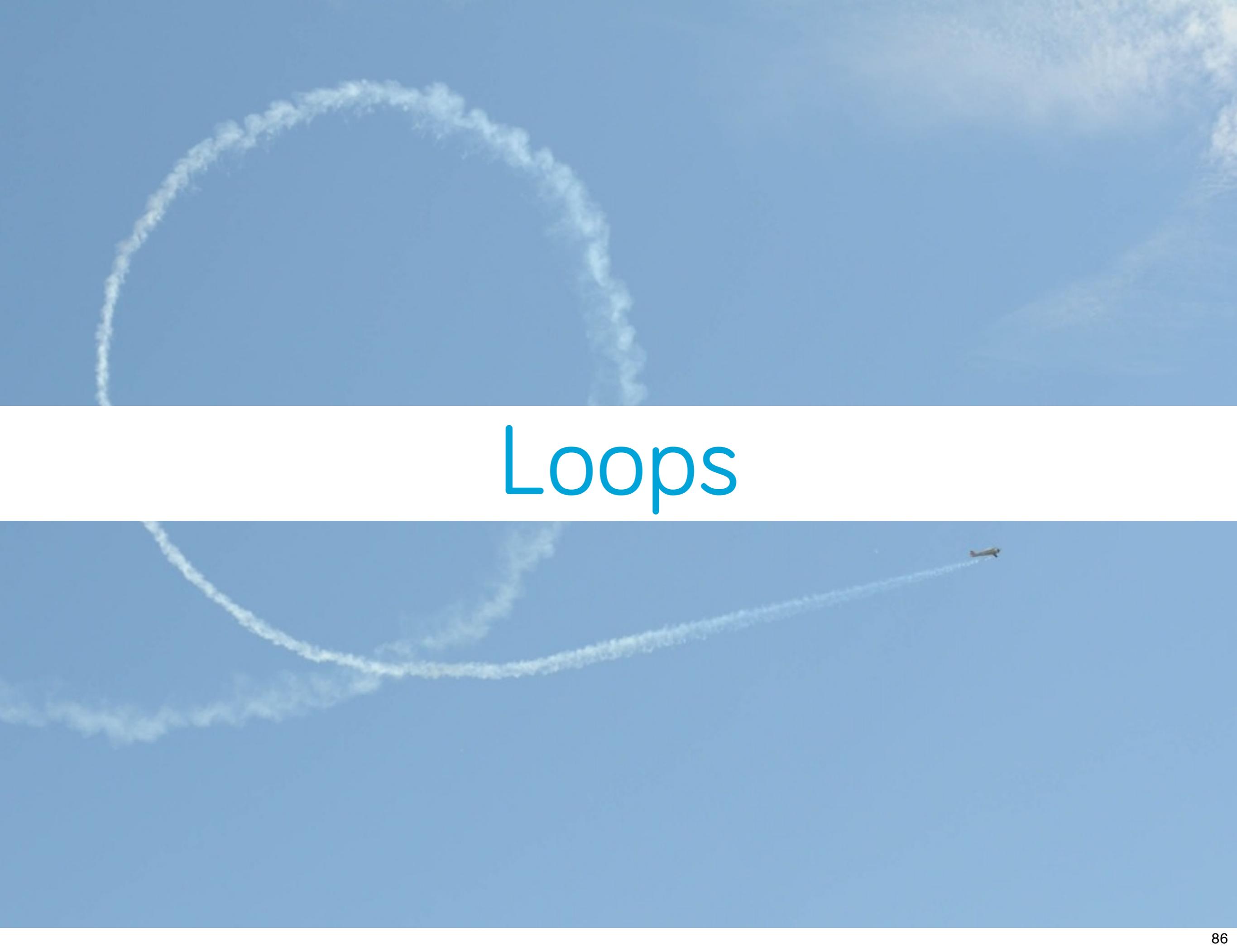
(They should have a width and height of 9.)

The image shows the Processing IDE interface. At the top is a toolbar with various icons. Below it is a menu bar with 'STANDARD' selected. The main area contains a code editor with the following content:

```
int xPos = 5;
int yPos = 7;

ellipse(xPos, yPos, 9, 9);
ellipse(xPos+10, yPos, 9, 9);
ellipse(xPos+40, yPos, 9, 9);
```

Below the code editor is a preview window titled 'sketc...'. Inside the window, there are two small black circles positioned at the top-left and center of a gray square.

A large, white, billowing smoke loop is centered in the upper portion of the image against a clear blue sky. The loop is roughly circular with some irregular edges. In the lower portion of the image, a smaller, thin white smoke trail extends from the left side towards the right, ending with a small, dark aircraft at its tip.

Loops



Loops

Humans

Far out in the uncharted backwaters of the unfashionable end of the Western Spiral arm of the Galaxy lies a small unregarded yellow sun.

Orbiting this at a distance of roughly ninety-eight million miles is an utterly insignificant little blue-green planet whose ape-descended life forms are so amazingly primitive that they still think digital watches are a pretty neat idea.

This planet has—or rather had—a problem, which was this: most of the people living on it were unhappy for pretty much of the time. Many solutions were suggested for this problem, but most of these were largely concerned with the movements of small green pieces of paper, which is odd because on the whole it wasn't the small green pieces of paper that were unhappy.

And so the problem remained; lots of the people were mean, and most of them were miserable, even the ones with digital watches.

Processing

The screenshot shows the Processing IDE interface. The code window contains the following Java code:

```
String name = "Jane";
name = "Rebecca";
println(name);
```

The output window below shows the result of the code execution:

```
Rebecca
```

... except when we
make it loop!



Loops

- Sections of code that are run multiple times, e.g.:
 - Every second
 - While a condition is true or false
 - A set number of times



Basic Loop

Pen & Paper Animation:

- First step: get together materials (pens, paper, paints, etc.)
- Second step: draw a frame on first piece of paper
- Third step: get a new piece of paper and draw the next frame
- Repeat until finished



Basic Loop

Pen & Paper Animation:

- First step: get together materials (pens, paper, paints, etc.)
- Second step: draw a frame on first piece of paper
- Third step: get a new piece of paper and draw the next frame
- Repeat until finished

Processing:

- First step: Processing looks at variables and their initial values
- Second step: Processing runs through certain code from top to bottom
- Third step: Processing goes back up to the top of that code and runs again
- Repeat until user closes program



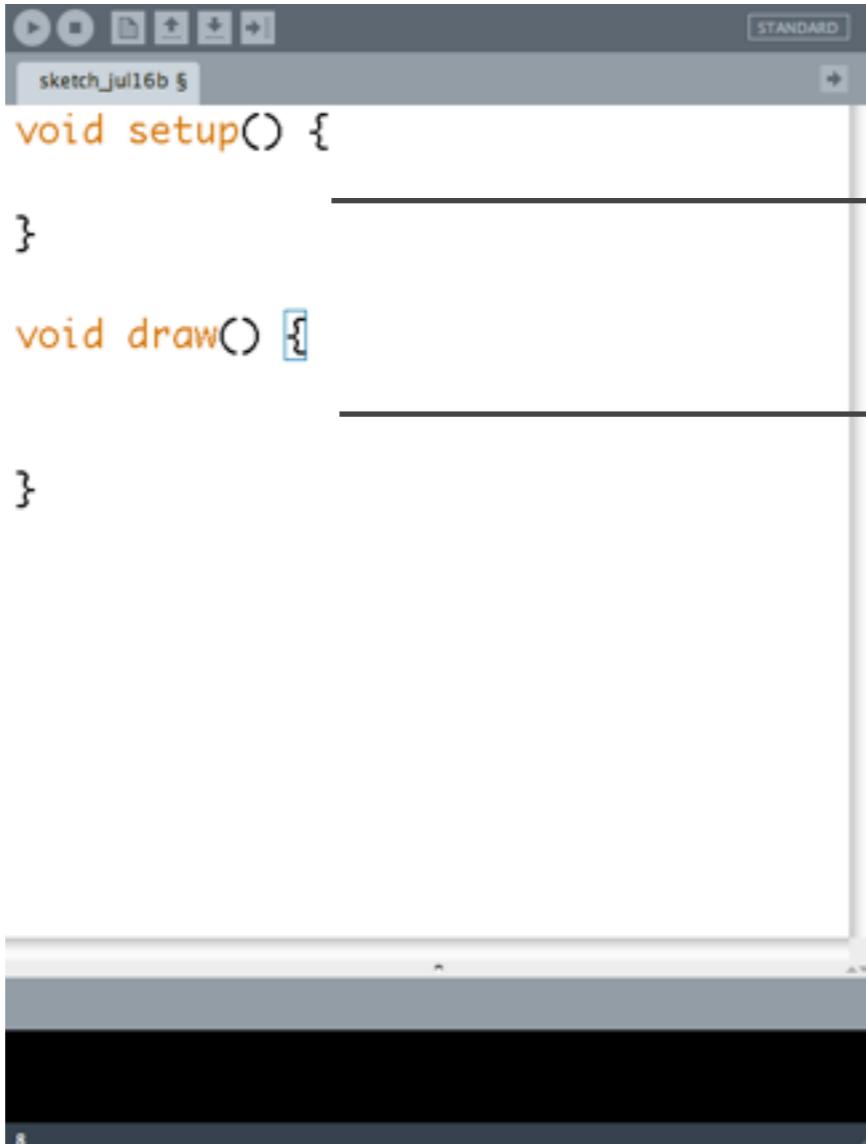
Basic Loop

Processing:

- First step: Processing looks at variables and their initial values → **setup**
 - Second step: Processing runs through certain code from top to bottom
 - Third step: Processing goes back up to the top of that code and runs again
- 
- draw**



Basic Sketch



```
sketch_jul16b § STANDARD
void setup() {
}
void draw() {
```

Two arrows point from the text "Where you set initial values." and "Where you write instructions to be repeatedly executed." to the "setup()" and "draw()" blocks respectively in the code editor.

Where you set
initial values.

Where you
write
instructions to
be repeatedly
executed.



Basic Sketch

The screenshot shows the Processing IDE interface with a sketch titled "sketch_jul17a". The code editor contains the following pseudocode:

```
void setup() {
    println("Hello");
}

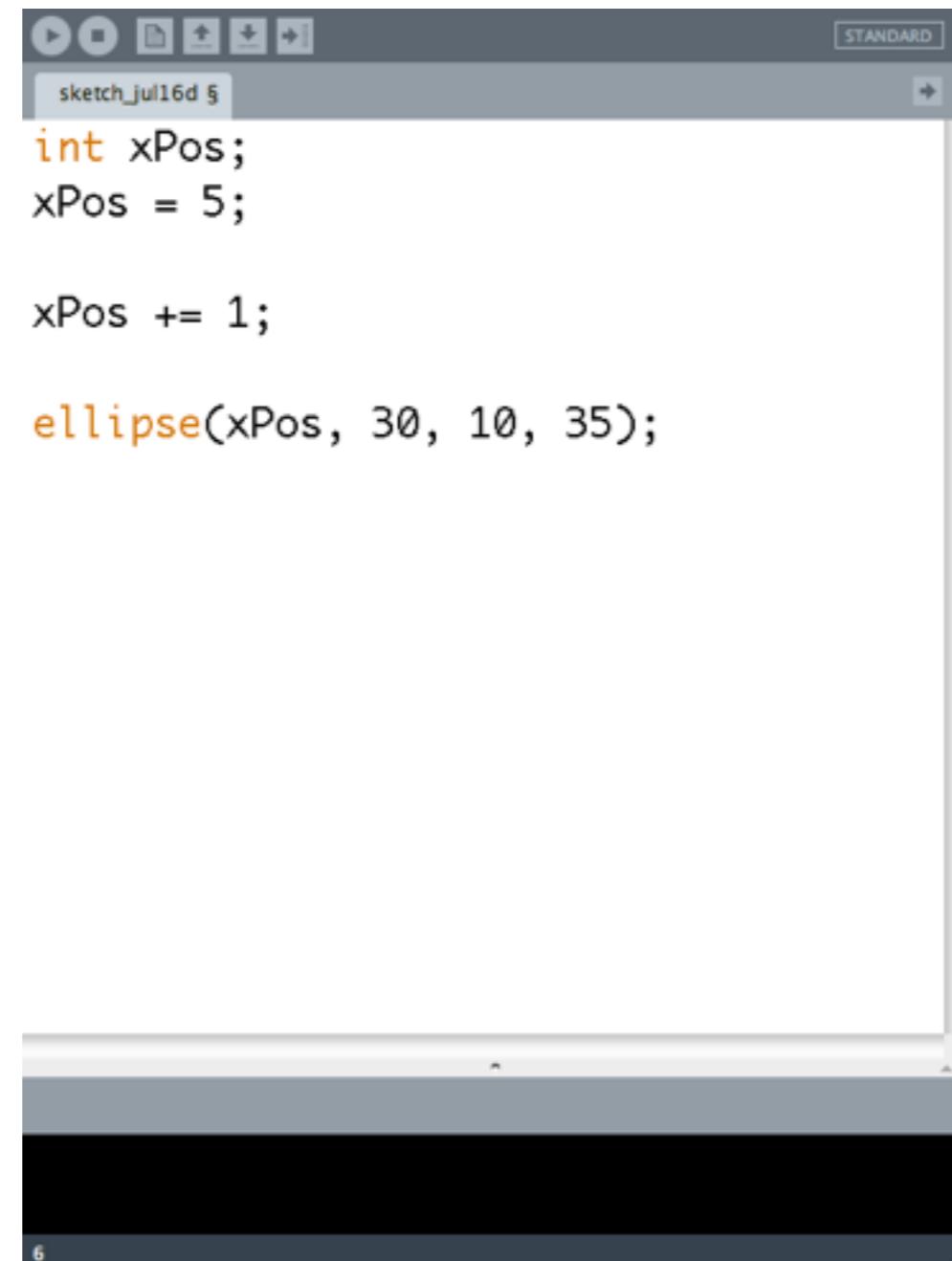
void draw() {
    println("1");
    println("2");
    println("3");
    println("4");
}
```

A black arrow points from the first line of the `setup()` function to the text "Runs first thing--but just once." A large black brace on the right side of the code editor spans from the end of the `draw()` function down to the bottom of the code editor window. To the right of this brace is the text "Goes top to bottom, then loops back up to top." A red brace at the bottom right points to the terminal window, which displays the output:

```
Hello
1
2
3
4
1
2
z
```

How would we arrange the following code so that Processing would add 1 to xPos every frame, and draw the ellipse at that new position?

What would go in setup, and what would go in draw?



```
sketch_jul16d 5
STANDARD
int xPos;
xPos = 5;

xPos += 1;

ellipse(xPos, 30, 10, 35);
```

```
int xPos;

void setup() {
  xPos = 5;
}

void draw() {
  xPos += 1;
  ellipse(xPos, 30, 10, 35);
}
```

xPos begins at 5, so we put that in setup.

Since we want to repeatedly add 1 to xPos, we put it in draw--that way, that code is run every frame.



Resulting Questions

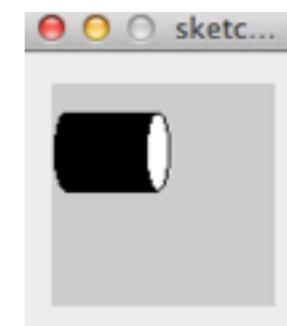
```
int xPos;

void setup() {
  xPos = 5;
}

void draw() {
  xPos += 1;
  ellipse(xPos, 30, 10, 35);
}
```

Why do we define `xPos` ("`int xPos`") outside of `setup`?

Why do we get those weird black lines?



Frame Systems





Basic Sketch

- Background function 'paints' over everything that was drawn previously
- Allows you to create actual animations/sense of motion

```
int xPos;

void setup() {
  xPos = 5;
}

void draw(){
  background(255);
  xPos += 1;
  ellipse(xPos, 30, 10, 35);
}
```



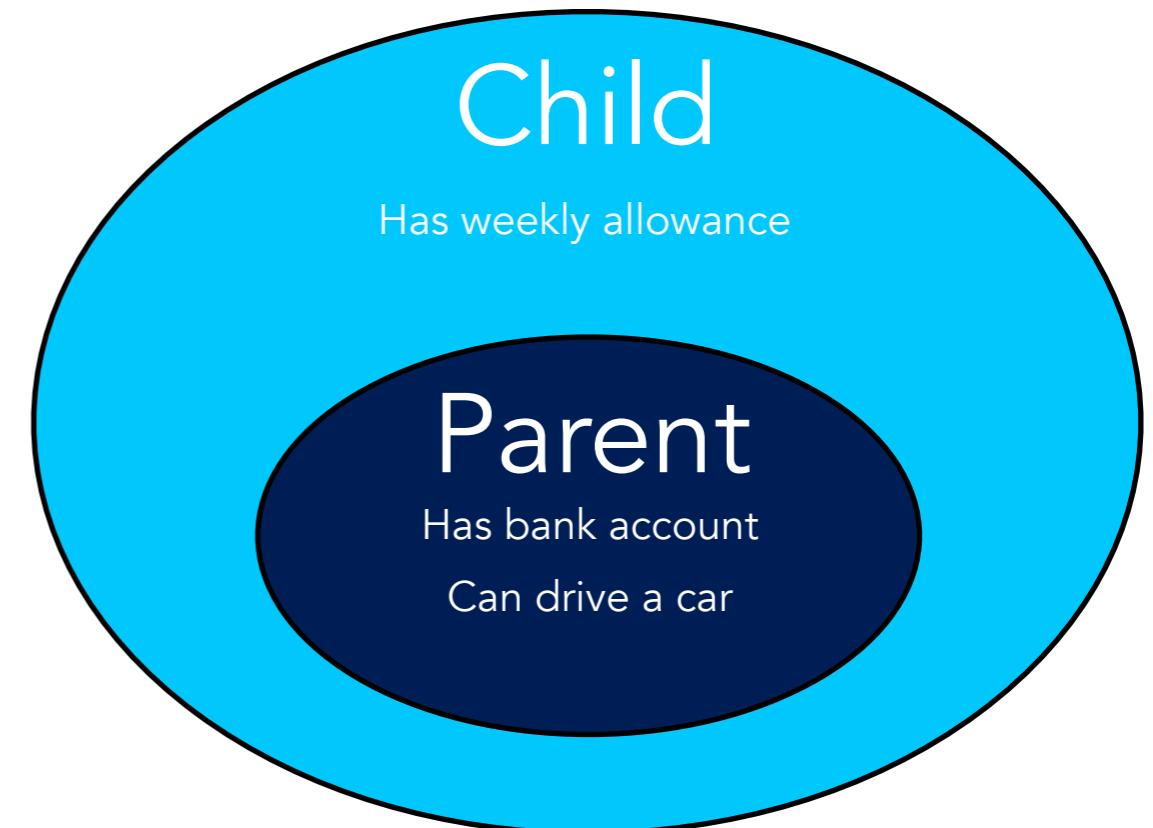
Scope



Scope

In real life:

- Parents may have a bank account and can drive cars
- Children may have allowances
- Parents can affect the child's allowance, but the children can't affect the parent's bank account or driving privileges

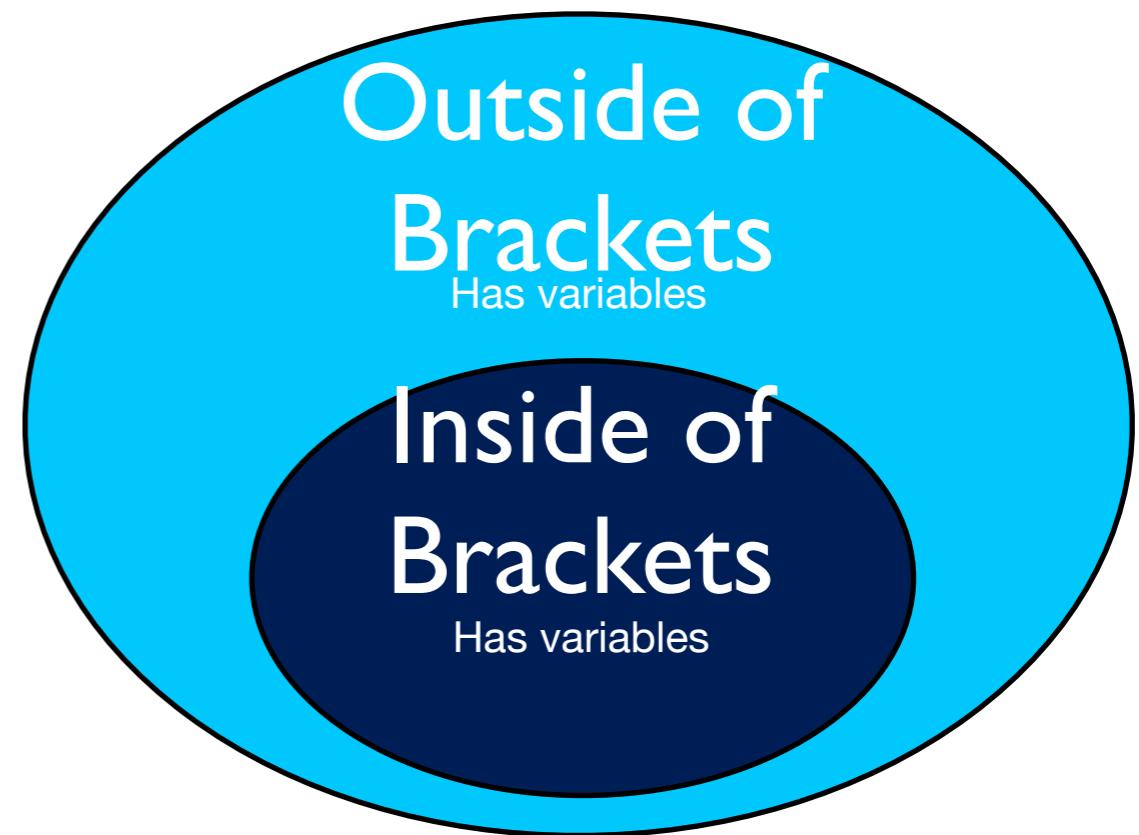




Scope

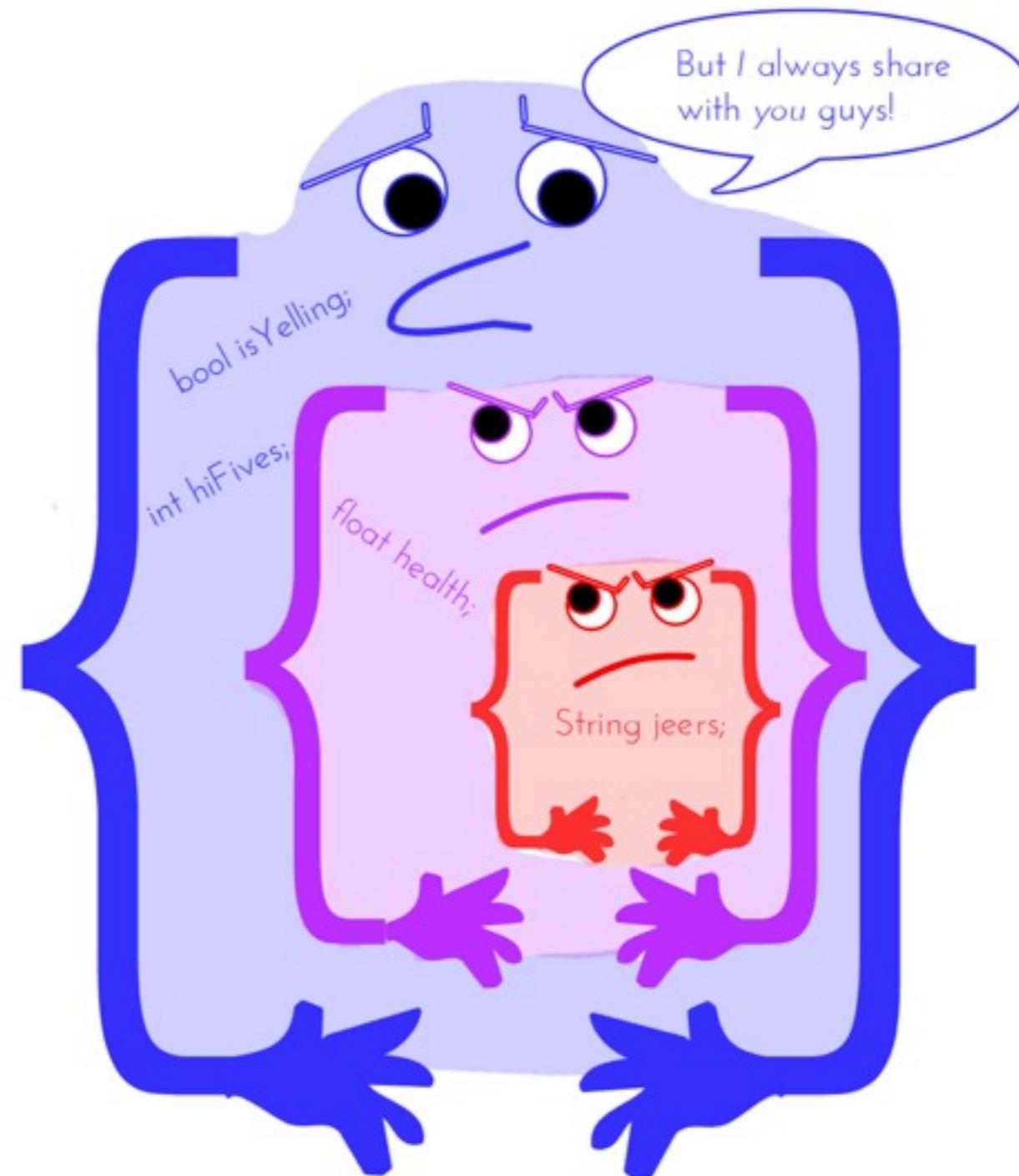
In programming (generally):

- Stuff inside a set of brackets can:
 - Create own variables
 - Interact with variables created outside brackets
- Stuff outside brackets can't affect variables created inside brackets





Scope





Scope

```
sketch_jul16a §
STANDARD
void setup() {
  int xPos = 5;
}

void draw() {
  background(255);
  xPos += 1;
  ellipse(xPos, 30, 10, 35);
}
```

xPos is defined within setup's brackets, meaning nothing outside setup can use it

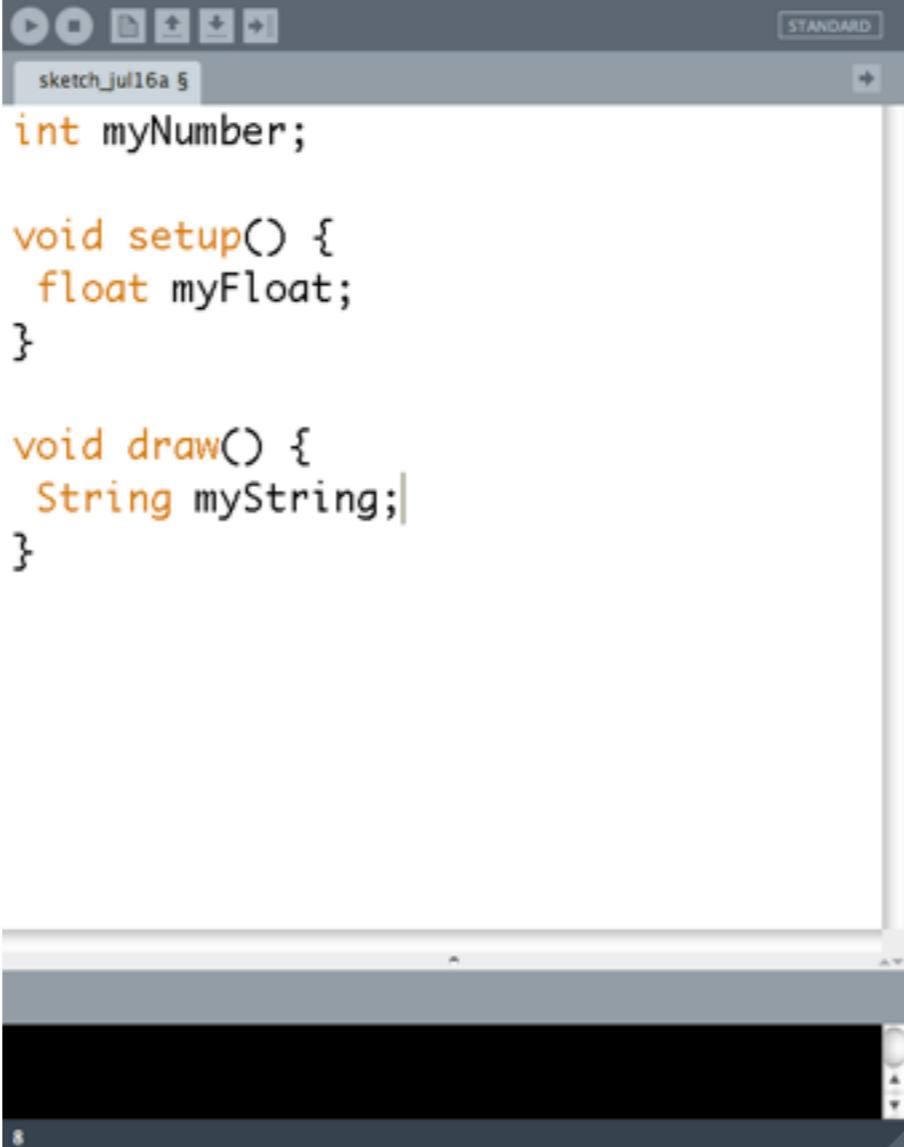
```
sketch_jul16c §
STANDARD
int xPos;

void setup() {
  xPos = 5;
}

void draw() {
  xPos += 1;
  ellipse(xPos, 30, 10, 35);
}
```

xPos is defined outside of setup and draw, so both of them can access and affect xPos

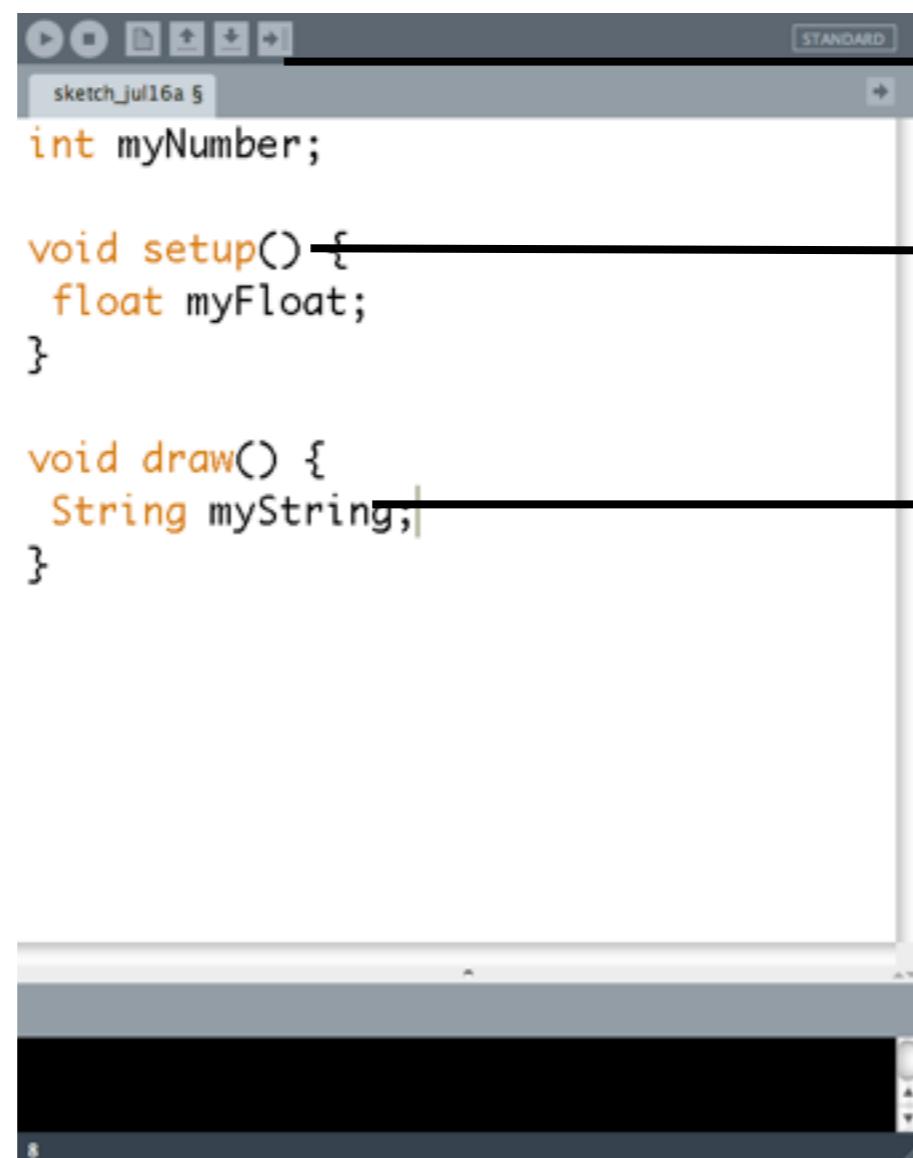
What parts of the code can access
myNumber? myFloat? myString?



```
sketch_jul16a § STANDARD
int myNumber;

void setup() {
  float myFloat;
}

void draw() {
  String myString;
}
```



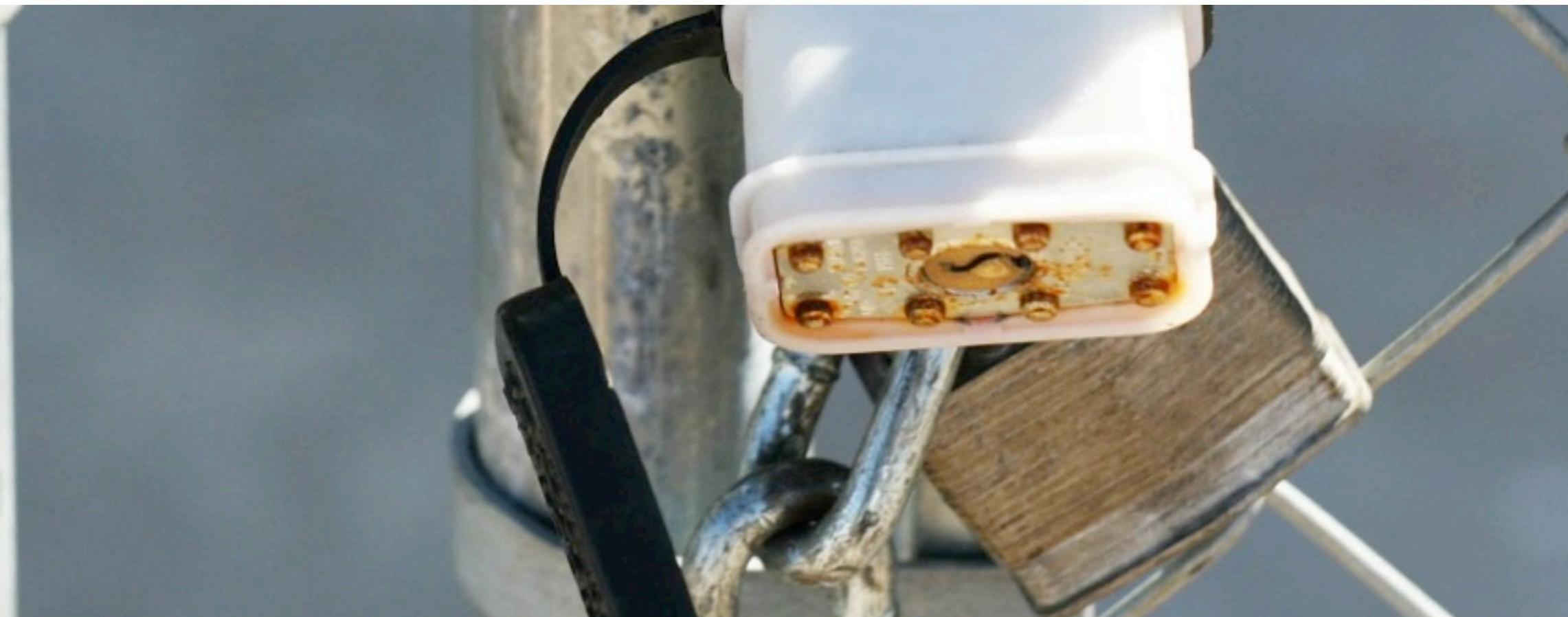
The image shows a screenshot of the Arduino IDE. At the top, there's a toolbar with various icons. Below it is a menu bar with 'FILE', 'SKETCH', 'TOOLS', 'HELP', and a 'STANDARD' button. The main area contains the code for 'sketch_jul16a'. The code includes a global variable declaration 'int myNumber;', a local variable declaration 'float myFloat;' within the 'setup()' function, and another local variable declaration 'String myString,' within the 'draw()' function.

```
int myNumber;  
  
void setup() {  
    float myFloat;  
}  
  
void draw() {  
    String myString;  
}
```

Annotations with arrows point from the code to the right:

- An arrow points from the 'myNumber' declaration to the text 'Global variable'.
- An arrow points from the 'myFloat' declaration to the text 'Local variable'.
- An arrow points from the 'myString' declaration to the text 'Local variable'.

Special Variables





Special Variables

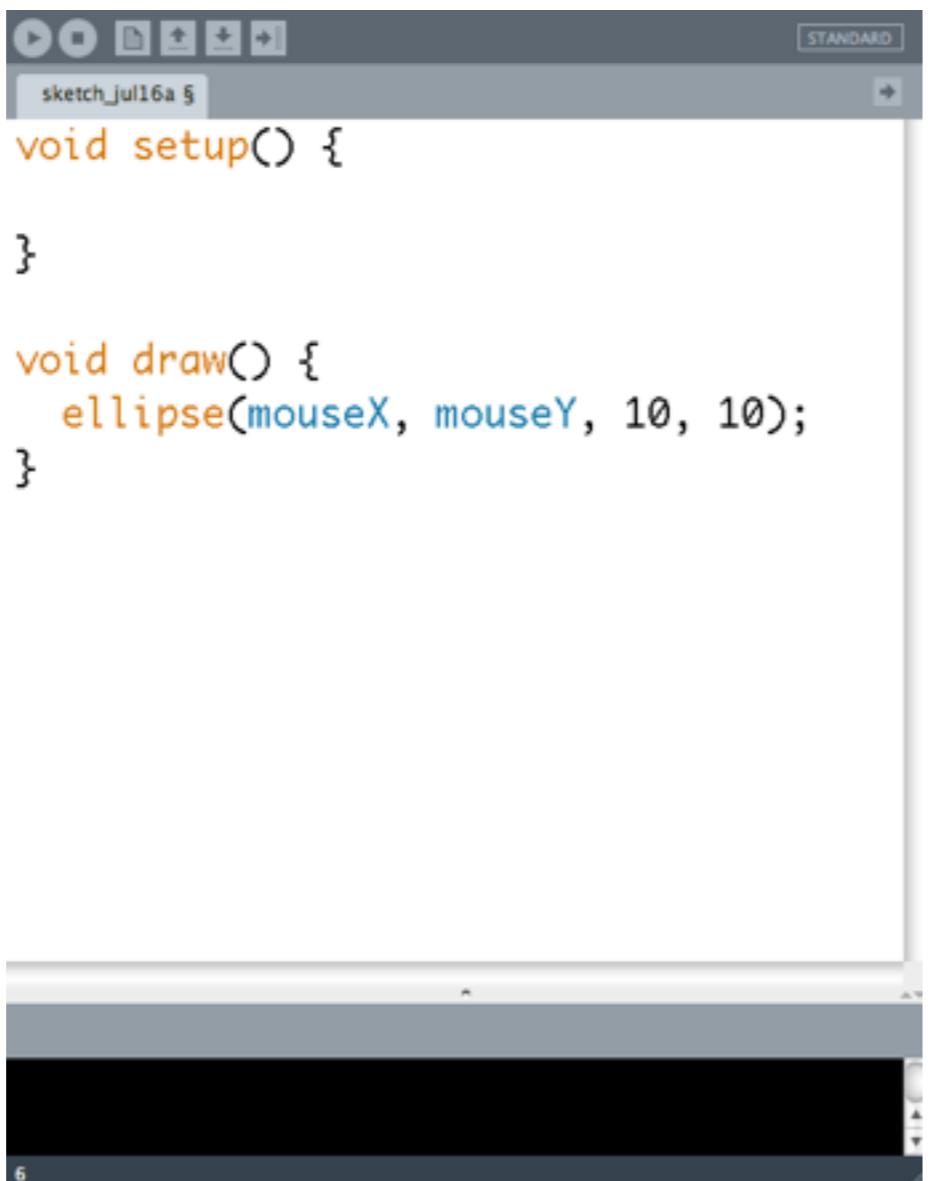
- There are certain variables where Processing keeps track of the values for us--we don't have to set them

Name	Use
<i>mouseX</i>	Current x-position of the mouse
<i>mouseY</i>	Current y-position of the mouse
<i>pmouseX</i>	X-position of the mouse one frame ago
<i>pmouseY</i>	Y-position of the mouse one frame ago
<i>width</i>	Width in pixels of the window
<i>height</i>	Height in pixels of the window

(these are just a few!)

Can you write code so that Processing draws a circle of width=10 and height=10 at the x and y position of the mouse every frame?

*Hint: the command for an ellipse is
ellipse(x-position, y-position, width, height);*



```
void setup() {  
}  
  
void draw() {  
    ellipse(mouseX, mouseY, 10, 10);  
}
```



If you wanted to get fancy, you could add in a `background()` command to `draw()`. Can you guess what that would do?



Let's Make A Simple Drawing App!

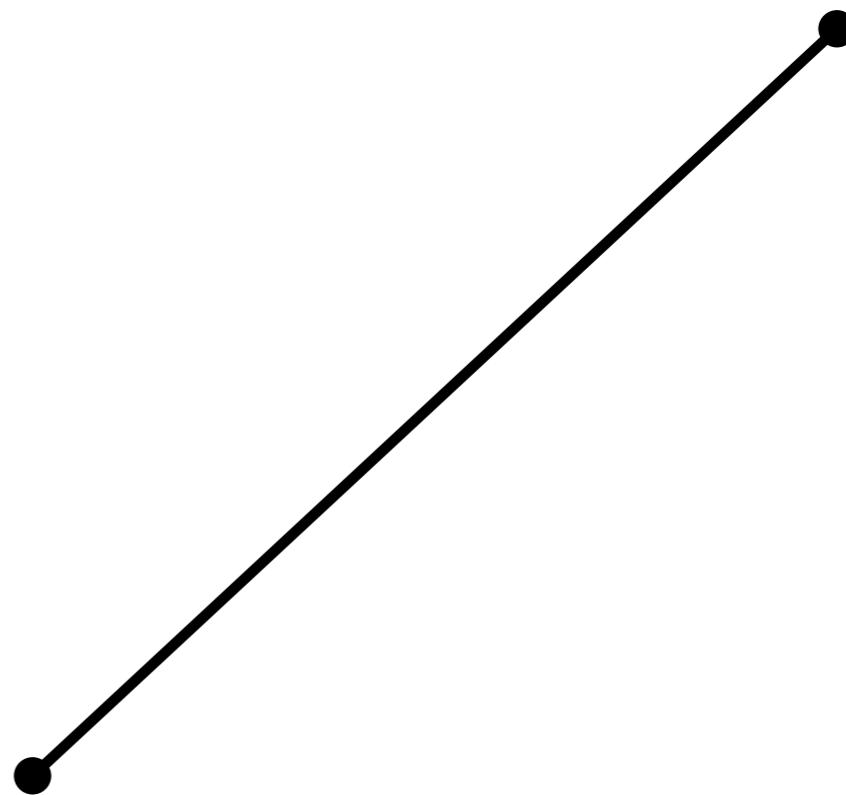




Simple Drawing App

In Processing, a line has two properties:

- A starting point (x_1, y_1)
- An ending point (x_2, y_2)



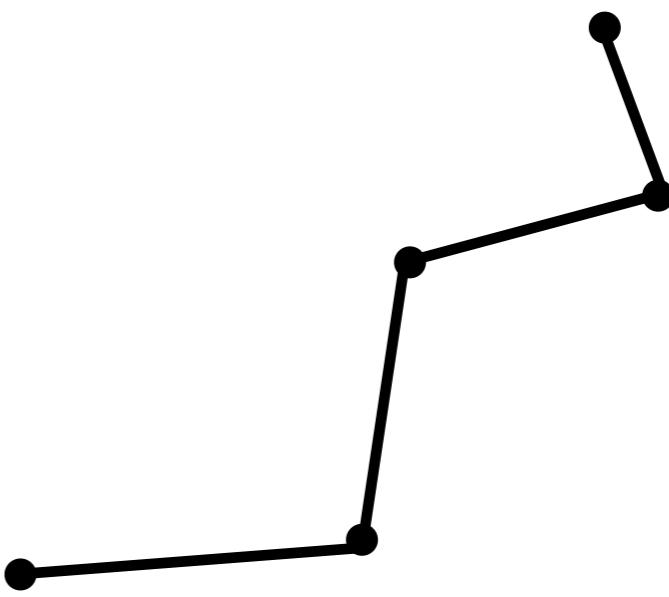


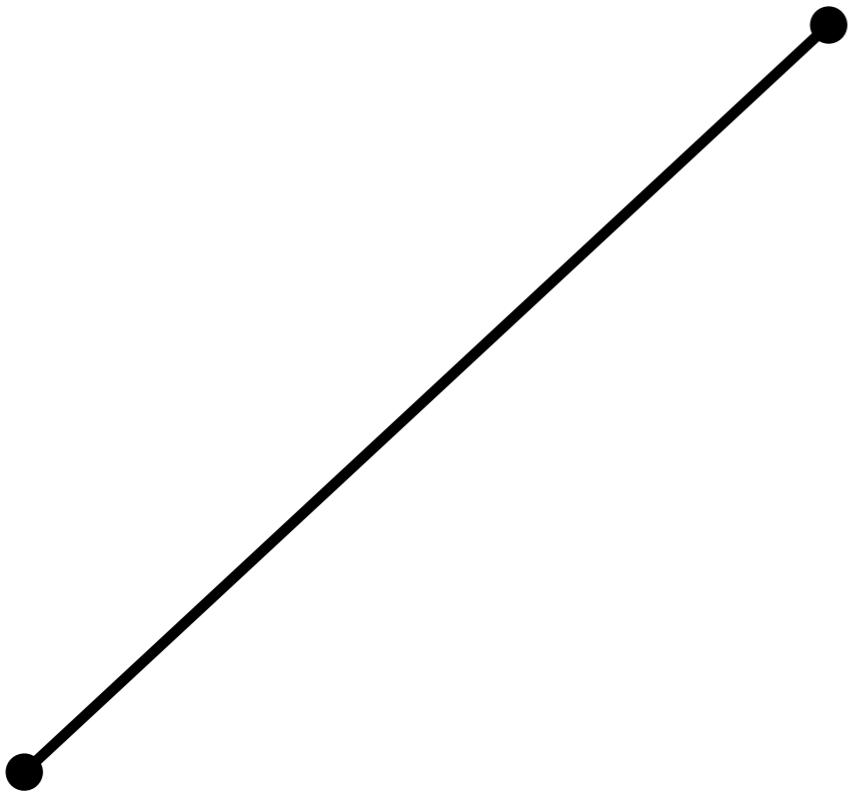
Simple Drawing App

In a wiggly line, the same idea holds. The line joining each point is made up of:

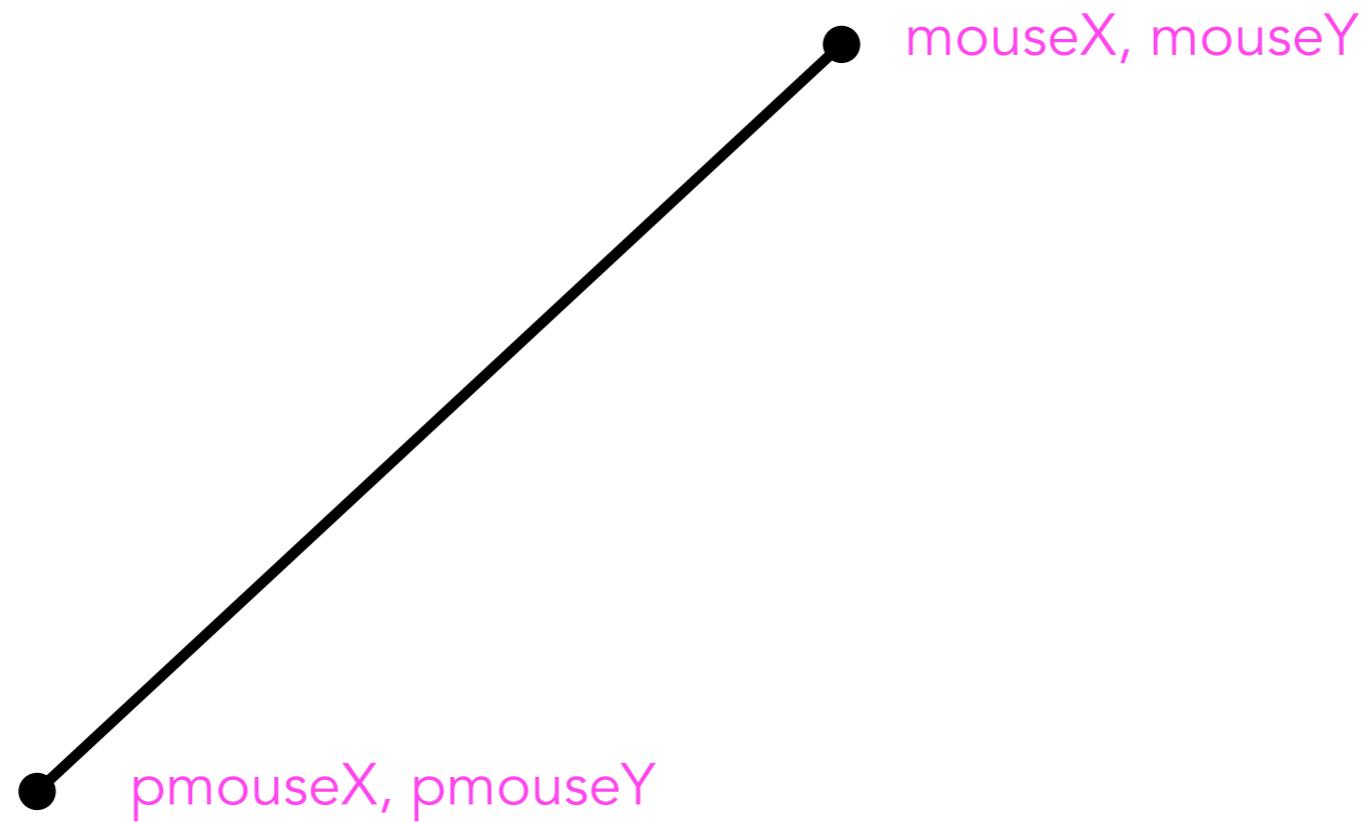
- A starting point (x, y)
- An ending point (x, y)

The only difference: the ending point of one part of the line becomes the starting point for the next part of the line.





Say our mouse started at the bottom-left point
and is now at the upper-left point. Which
Processing variables would describe each point?



What would happen if, every frame, we drew a line between where the mouse was last frame, and where the mouse is now?

Can you write this code?

The image shows the Processing IDE interface. At the top, there's a toolbar with various icons for file operations like Open, Save, and Print. To the right of the toolbar is a "STANDARD" button. Below the toolbar, the title bar displays "sketch_jul16a §". The main area contains the following code:

```
void setup() {  
}  
  
void draw() {  
    line(pmouseX, pmouseY, mouseX, mouseY);  
}
```

Below the code editor is a preview window titled "sketc...". Inside the preview window, there is a small gray square containing a simple line drawing of a hand holding a pen, which has drawn a few curved lines on the canvas.

Conditionals



You must be at least this tall to read these books



Conditionals

- A conditional statement is a piece of code that only runs in certain cases/depending on certain conditions
- E.g.: you must be of a certain height to ride a rollercoaster--if you're less than that height, you can't ride it



Conditionals

If your height > the minimum height => you can ride

Otherwise, you won't be able to ride it



Conditionals

If-statement

```
if (var1 > var2) {  
    println("var1 is greater than var2");  
}
```

- Condition that needs to be met for bracketed code to execute.
- Resulting action if condition is met.
- Open and closed brackets indicate to computer what is actually part of the conditional statement.

If-Else Statement

```
if (var1 > var2) {  
    println("var1 is greater than var2");  
} else {  
    println("var1 is not greater than var2");  
}
```

- Condition that needs to be met for bracketed code to execute.
- Resulting action if condition is met.
- If the condition is not met...
- The resulting action if the condition is not met.



Comparators

Operator	Use	Examples
Greater than (>)	Checking to see if something is larger than something else	<pre>if (myAge > 25) { canRentCar = true; }</pre>
Less than (<)	Checking to see if something is smaller than something else	<pre>if (myAge < 25) { canRentCar = false; }</pre>
Greater than or equal to (>=)	Checking to see if something is larger than or equal to something else	<pre>if (myAge >= 21) { canEnterBar = true; }</pre>
Less than or equal to (<=)	Checking to see if something is smaller than or equal to something else	<pre>if (myAge <= 20) { canEnterBar = false; }</pre>



Comparators

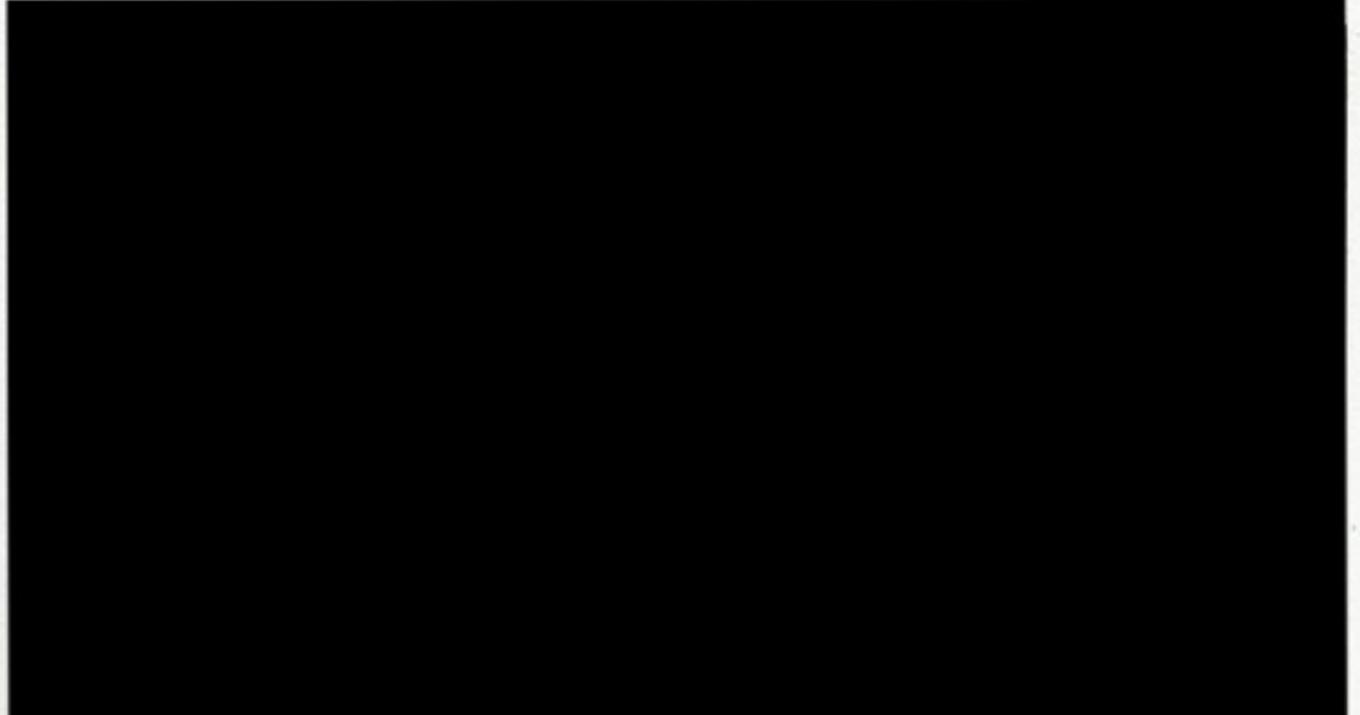
Operator	Use	Examples
Double-equals (==)	Checking to see if something equals something else	<pre>if (age == 16) { sweet_sixteen = true; }</pre>
Not>equals (!=)	Seeing if something is not the case.	<pre>if (happy != true) { mood = "bad"; }</pre>

Can you write the following code?

If the mouse is less than halfway across the sketch, make the background red. Otherwise, make the background blue.

If the mouse is less than halfway across the sketch, make the background red.
Otherwise, make the background blue.

```
void setup() {}  
void draw() {  
    if (mouseX < 50) {  
        background(255, 0, 0);  
    } else {  
        background(0, 0, 255);  
    }  
}  
// Default sketch size is 100, so 50 is the midpoint.
```



Images & Fonts





Images and Fonts

- Remember our earlier datatypes? (int, float, etc.)
- There are two more fun datatypes:
 - PImage (image)
 - PFont (font)



Images and Fonts

- You can declare image and font variables using these datatypes
- These datatypes also have *functions*--different actions they can perform on themselves

PImage

- The ability to load an image.
- The ability to resize an image.
- The ability to draw that image.

Pfont

- The ability to load a font.
- The ability to resize that font.
- The ability to write stuff in that font.



Images and Fonts

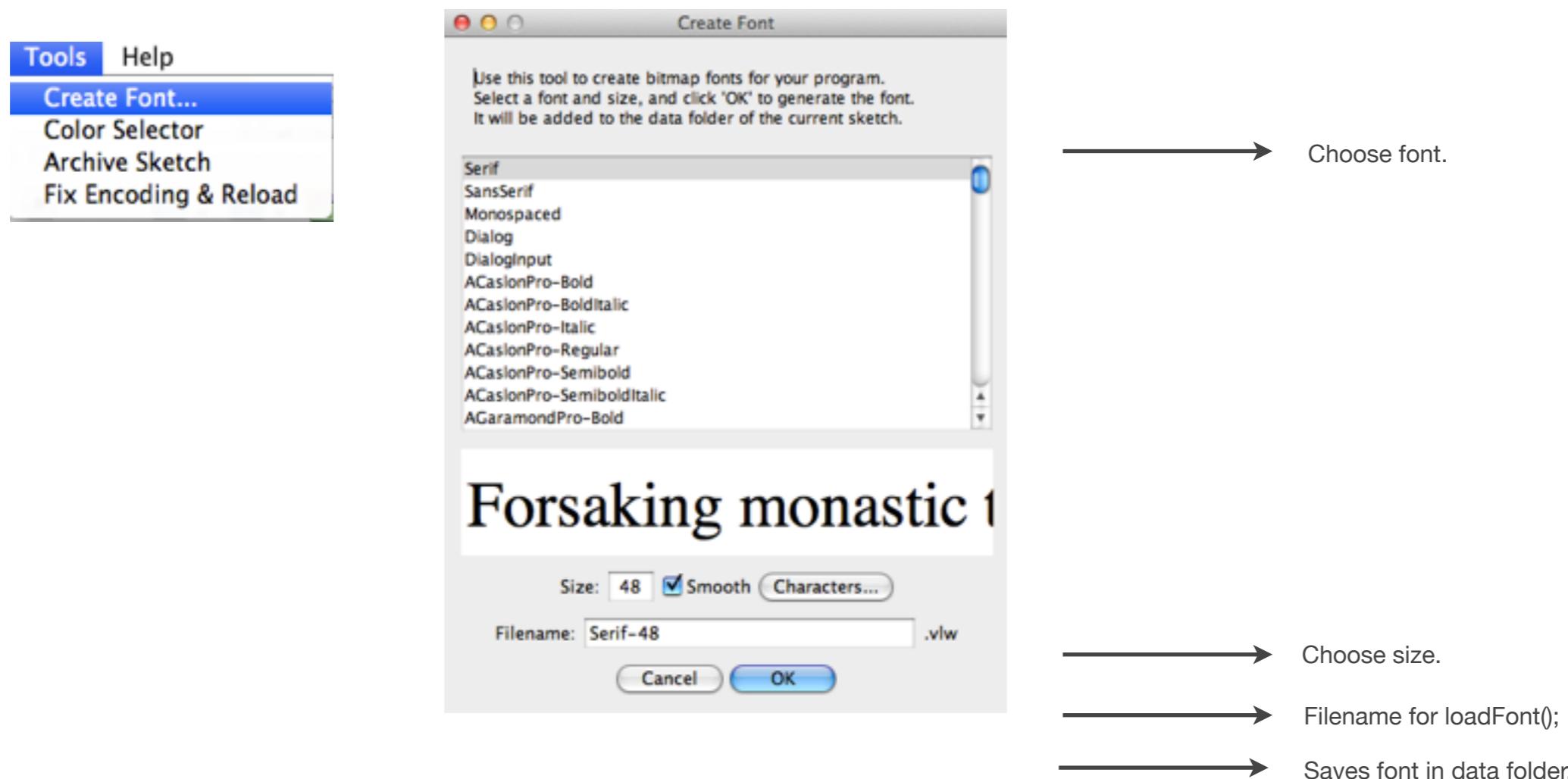
Step	Purpose	Sample Code
Defining the image	Telling Processing to set aside memory space to hold an image	<code>PImage p;</code>
Loading the image	Telling Processing the filename of this image so it can find and load it into that space	<code>p = loadImage("pic.png");</code>
Drawing the image	Telling Processing to draw that image at a given location, via the 'image' method	<code>image(p, xPos, yPos);</code>



Images and Fonts

Step	Purpose	Sample Code
Defining the font	Telling Processing to set aside memory space to hold a font	<code>PFont f;</code>
Creating the font	Picking a font and converting it to a form Processing can use	N/A--see next slide
Loading the font	Telling Processing the filename of this font so it can find and load it into that space	<code>f = loadFont("Arial.vlw");</code>
Using the font	Telling Processing to make all subsequent text this font at this size	<code>textFont(f, 32);</code>
Writing stuff	Actually writing things in that font	<code>text("Whatever!");</code>

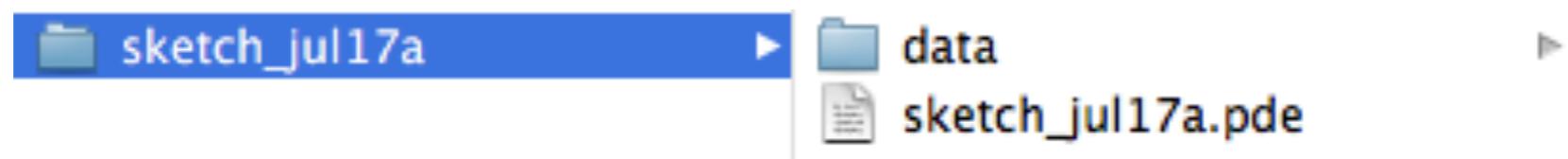
Loading Fonts





Images and Fonts

- Outside data, like images and fonts, are always stored in the 'data' folder of that sketch
- If you don't see one, just make it



Try downloading an image and drawing it in a sketch (at whatever location you like).

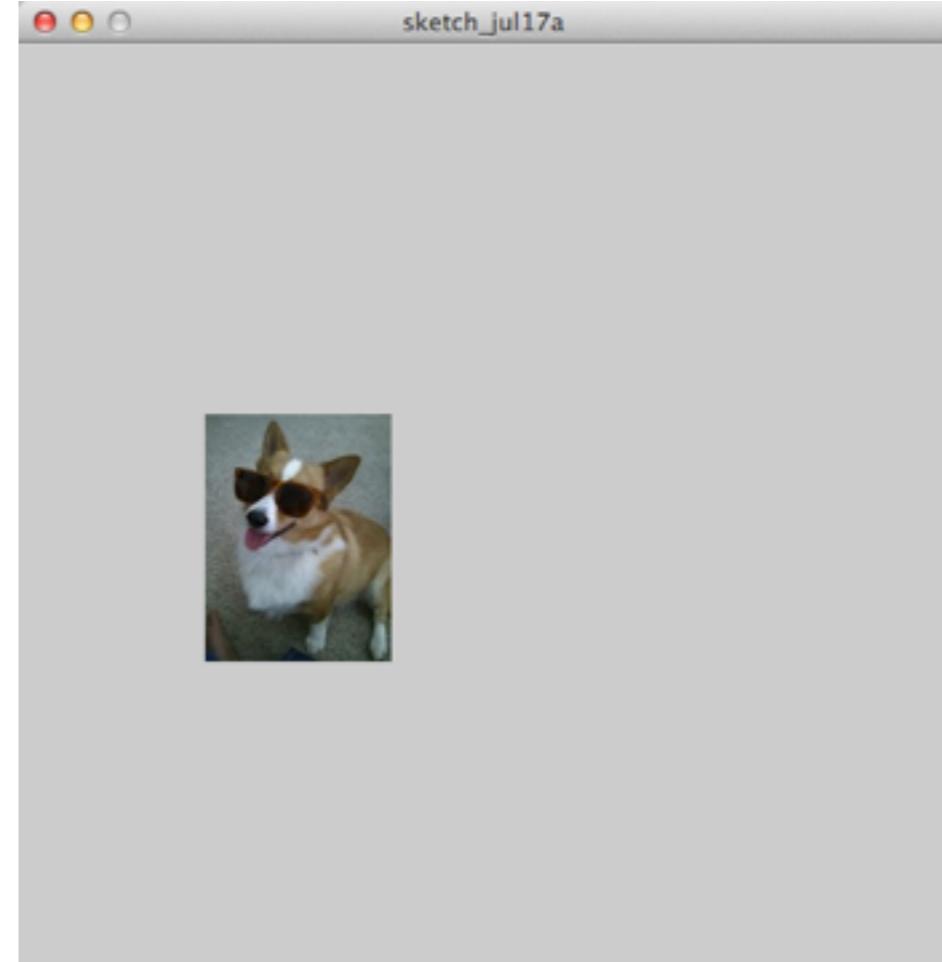
Hint: put the file in your 'data' folder. Define it outside of setup() and draw(), and load the image in setup().

(Do you know why we would load it in setup rather than draw())?

```
PImage corgi;

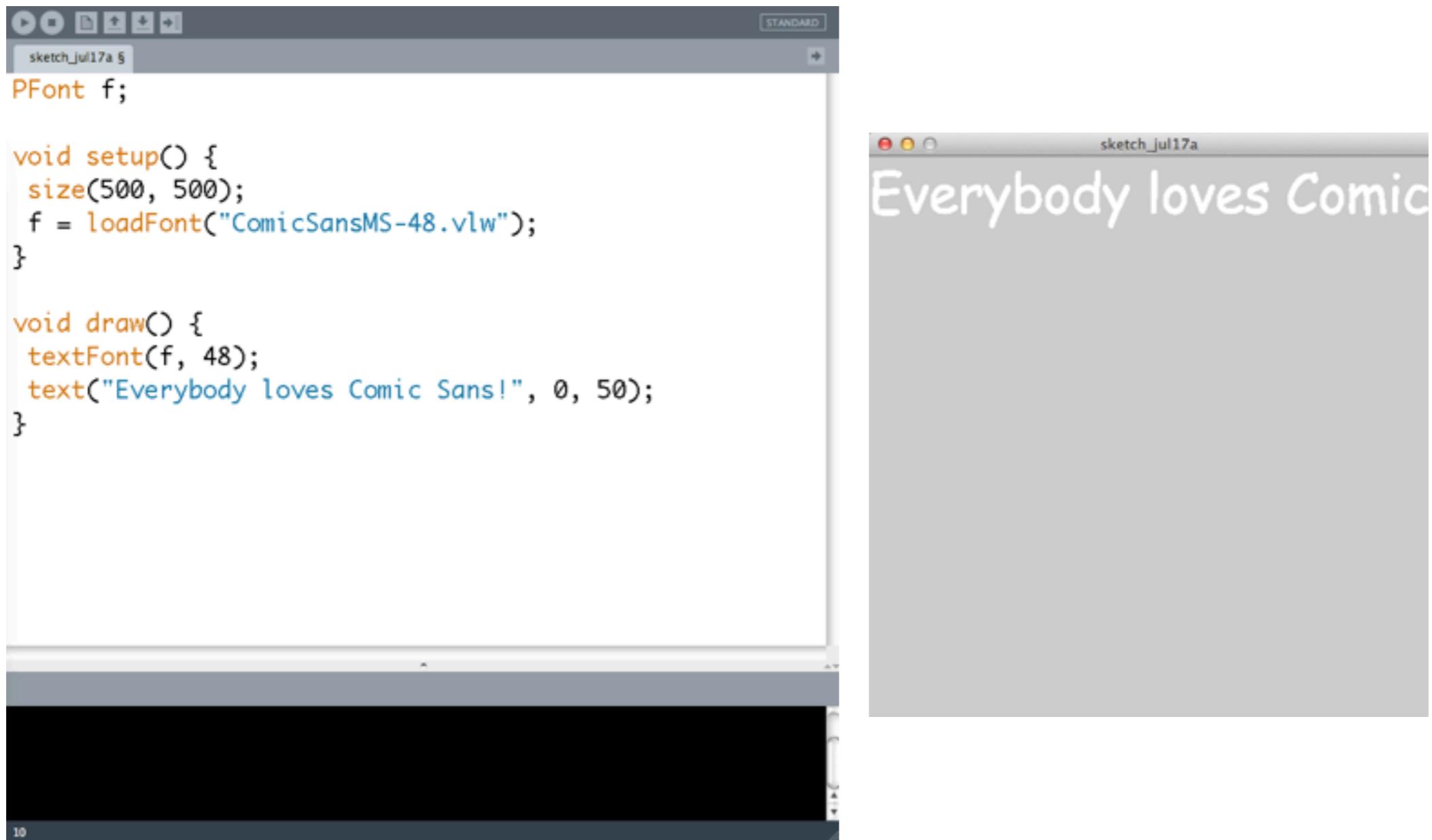
void setup() {
  size(500, 500);
  corgi = loadImage("corgi.jpg");
}

void draw() {
  image(corgi, 100, 200);
}
```



Try creating a font and using it in a sketch.

*Hint: to write text with a font, use this command at the end:
text("text", xPos, yPos);*

The image shows the Processing IDE interface. On the left, the code editor contains the following sketch code:

```
PFont f;  
  
void setup() {  
    size(500, 500);  
    f = loadFont("ComicSansMS-48.vlw");  
}  
  
void draw() {  
    textAlign(CENTER);  
    textFont(f, 48);  
    text("Everybody loves Comic Sans!", 0, 50);  
}
```

The code uses the `loadFont` function to load a font file named "ComicSansMS-48.vlw". In the `draw()` function, the text "Everybody loves Comic Sans!" is displayed in the center of the 500x500 pixel canvas using the loaded font.

On the right, a preview window titled "sketch_jul17a" displays the text "Everybody loves Comic Sans!" in a white, slightly italicized font, matching the appearance in the code editor.



Escape Characters

To force a new line/return, use \n.

The image shows the Processing IDE interface. On the left, the code editor contains the following sketch:

```
PFont f;  
  
void setup() {  
    size(500, 500);  
    f = loadFont("ComicSansMS-48.vlw");  
}  
  
void draw() {  
    textFont(f, 48);  
    text("Everybody loves\nComic Sans!", 0, 50);  
}
```

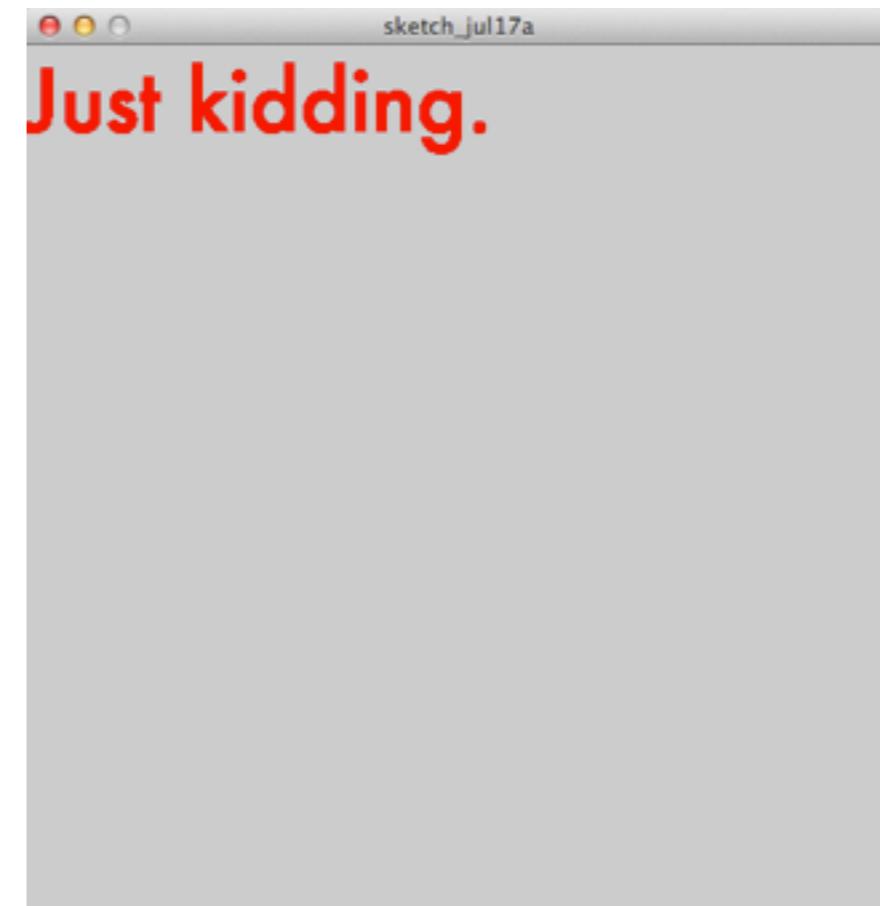
The code uses the `\n` escape character to create a new line in the text output. On the right, the preview window titled "sketch_jul17a" displays the text "Everybody loves" on the first line and "Comic Sans!" on the second line, demonstrating the effect of the escape character.

Colors

Try coloring the text you just wrote by using a
fill() command.

The screenshot shows the Processing IDE interface. The title bar says "sketch_jul17a". The code area contains the following:

```
PFont f;  
  
void setup() {  
    size(500, 500);  
    f = loadFont("Futura-Medium-48.vlw");  
}  
  
void draw() {  
    textFont(f, 48);  
    fill(255, 0, 0);  
    text("Just kidding.", 0, 50);  
}
```



Try the following exercises:

Write a sketch such that an image is drawn at the current location of the mouse.

Write a sketch such that when the mouse is on the left-hand side of the screen, text says 'left', and when it is on the right-hand side of the screen, text says 'right.'

Hint: both require use of background() in draw to look right.

```
sketch_jul17a § STANDARD +  
PFont f;  
  
void setup() {  
    size(500, 500);  
    f = loadFont("Futura-Medium-48.vlw");  
}  
  
void draw() {  
    background(0);  
    textAlign(CENTER, CENTER);  
    textSize(48);  
  
    if (mouseX <= 250) {  
        text("Left!", 100, 50);  
    } else {  
        text("Right!", 100, 50);  
    }  
}
```

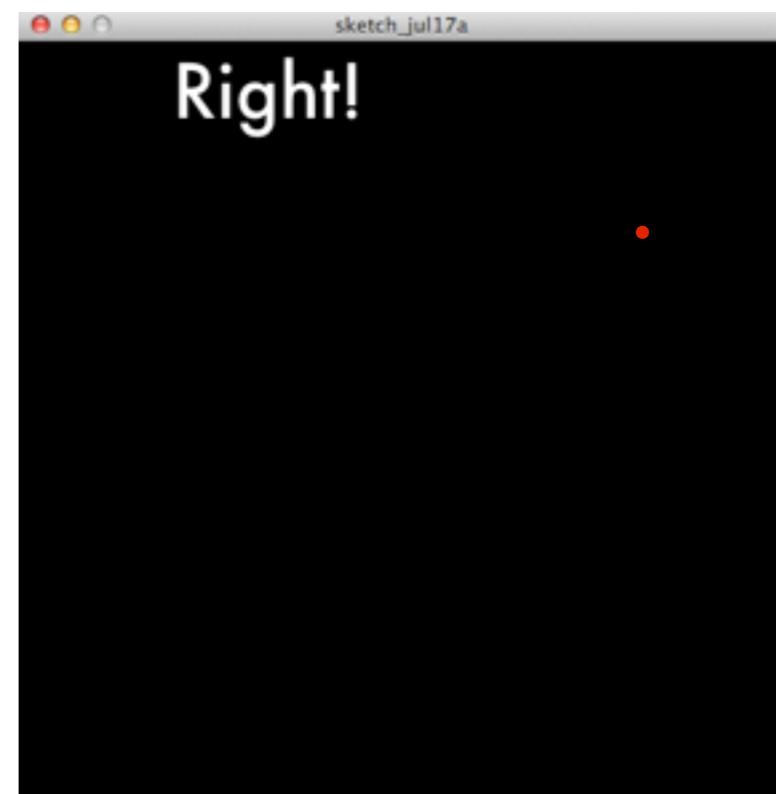




Image Mode

- By default, an image's x and y position are in the upper-left-hand corner

The screenshot shows the Processing IDE interface. The top bar displays the title "sketch_jul17a §" and the mode "STANDARD". The code area contains the following Pseudocode:

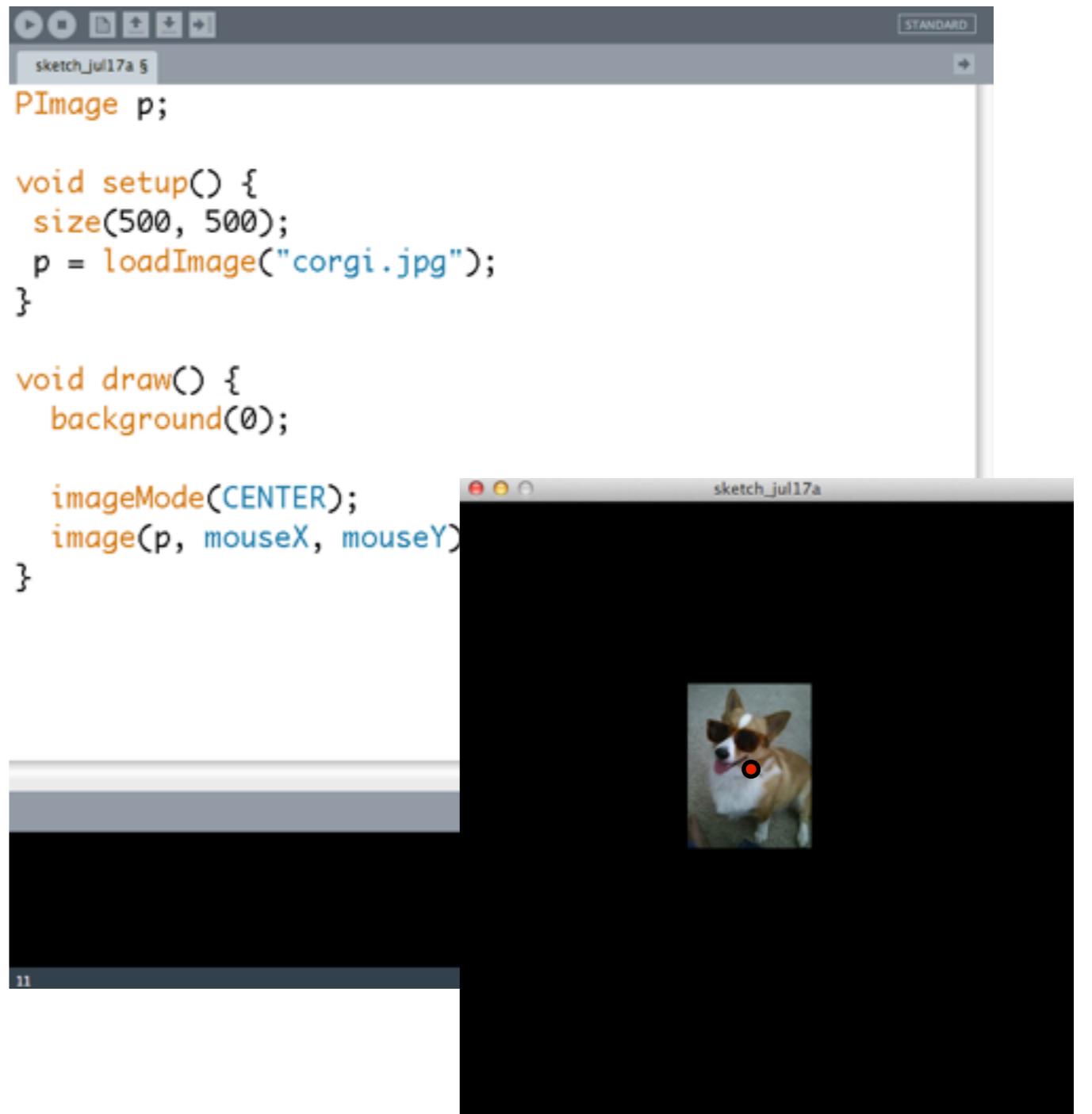
```
PImage p;  
  
void setup() {  
    size(500, 500);  
    p = loadImage("corgi.jpg");  
}  
  
void draw() {  
    background(0);  
  
    image(p, mouseX, mouseY);  
}
```

To the right of the code is a preview window titled "sketch_jul17a" showing a black canvas with a small image of a dog wearing sunglasses centered at the mouse cursor.



Image Mode

- You can use `imageMode(CENTER)` to move the x and y to the center



The screenshot shows the Processing IDE interface. The top bar displays the title "sketch_jul17a" and the mode "STANDARD". The code area contains the following pseudocode:

```
PImage p;  
  
void setup() {  
    size(500, 500);  
    p = loadImage("corgi.jpg");  
}  
  
void draw() {  
    background(0);  
  
    imageMode(CENTER);  
    image(p, mouseX, mouseY)  
}
```

The bottom right corner of the code editor shows a small thumbnail of a dog's face, which is the image being loaded. The bottom right corner of the slide also shows a larger version of the same dog's face.



SO R@n:D0Z!

Randomness

SO R@n:D0Z!

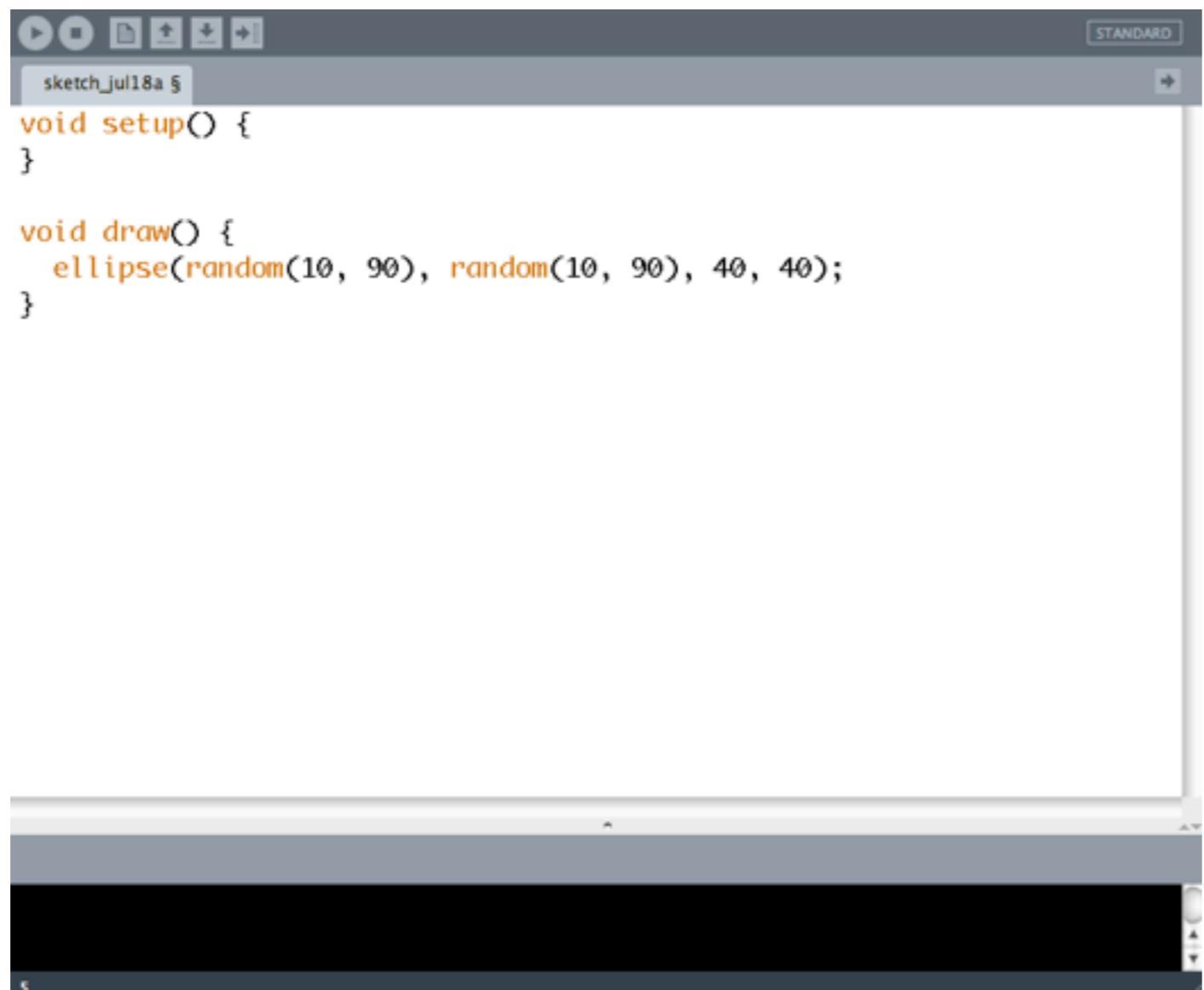


Randomness

- Useful for making things unpredictable!
- Takes either one or two parameters:
 - `random(5)` returns a number from 0->just under 5
 - `random(3, 5)` returns a number between 3->just under 5

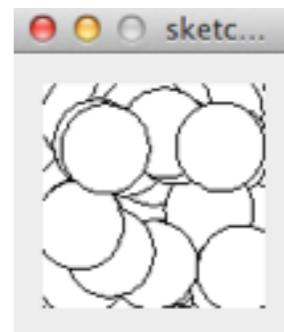
Try drawing an ellipse at a random x-position between 10 and 90, and a random y-position between 10 and 90.

Do this in setup(). Now try it in draw(). Why the difference?



```
sketch_jul18a § STANDARD
void setup() {
}

void draw() {
    ellipse(random(10, 90), random(10, 90), 40, 40);
}
```



A dynamic photograph of a football game. In the center, a player in a white jersey and green pants is in possession of the ball, leaning forward. To his right, a player in a black jersey and yellow pants is pushing him. Another player in a black jersey and yellow pants is partially visible behind the central player. On the far left, a player in a black jersey with '54' and yellow pants is running towards the camera. The background shows a grassy field and a line of trees.

How can I test for collisions?

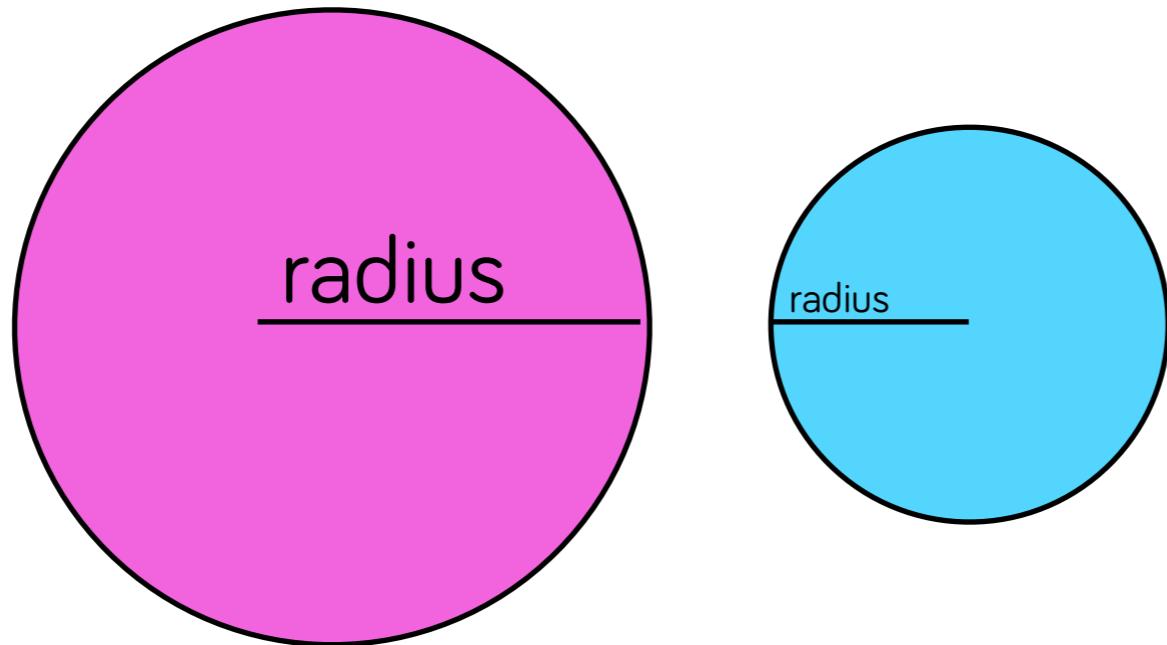


Distance

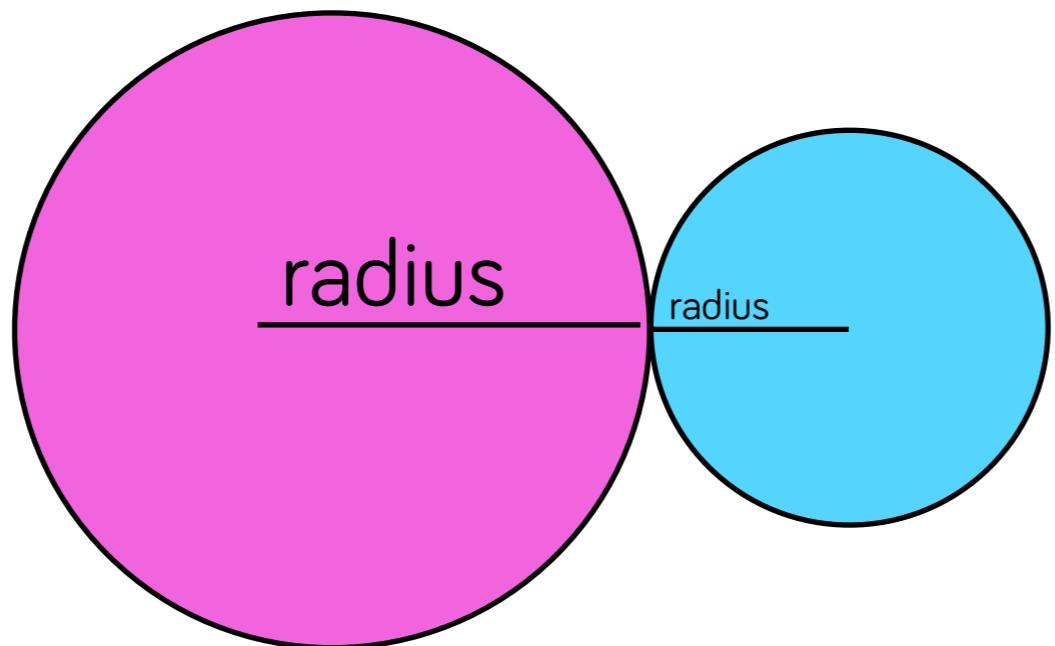
Collision: when one point is less than a certain distance from another point.



Have these circles collided yet?



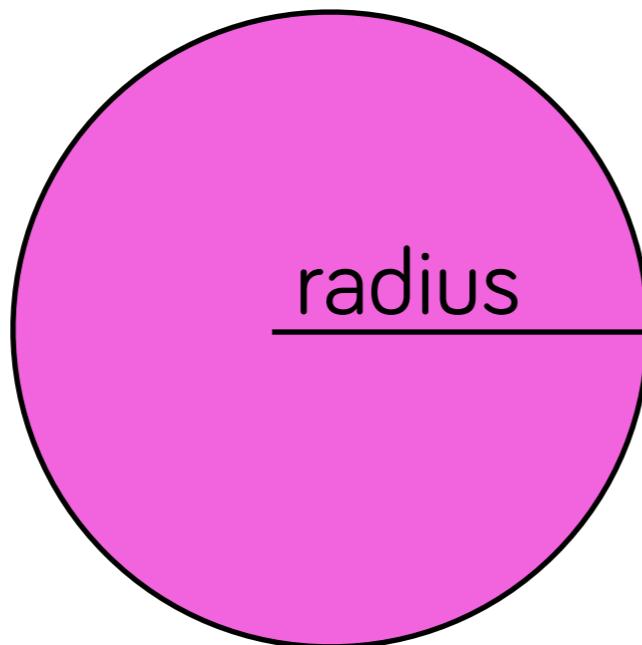
How about now?





Distance

- If the distance between the center-points of the circles is less than or equal to the sum of their radii, they have collided!
- Get distance with `dist(x1, y1, x2, y2)`





Distance

- If the distance between the center-points of the circles is less than or equal to the sum of their radii, they have collided!
- Get distance with `dist(x1, y1, x2, y2)`

