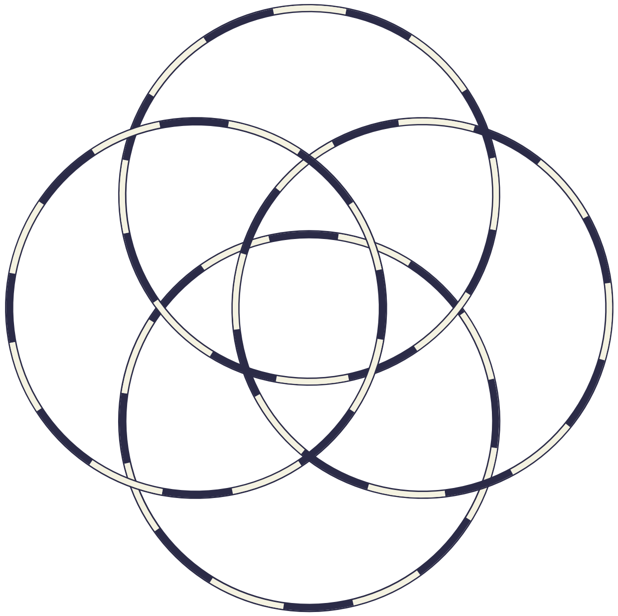
# PROCESSING INTRODUCTION

This white side will become illustrations when we do concepts more abstract.

Typing isn't too abstract so instead enjoy whatever this is :)



## PRINTING TO THE CONSOLE

On the first blank line, type:
print("Hello World!");
or
print(a number);

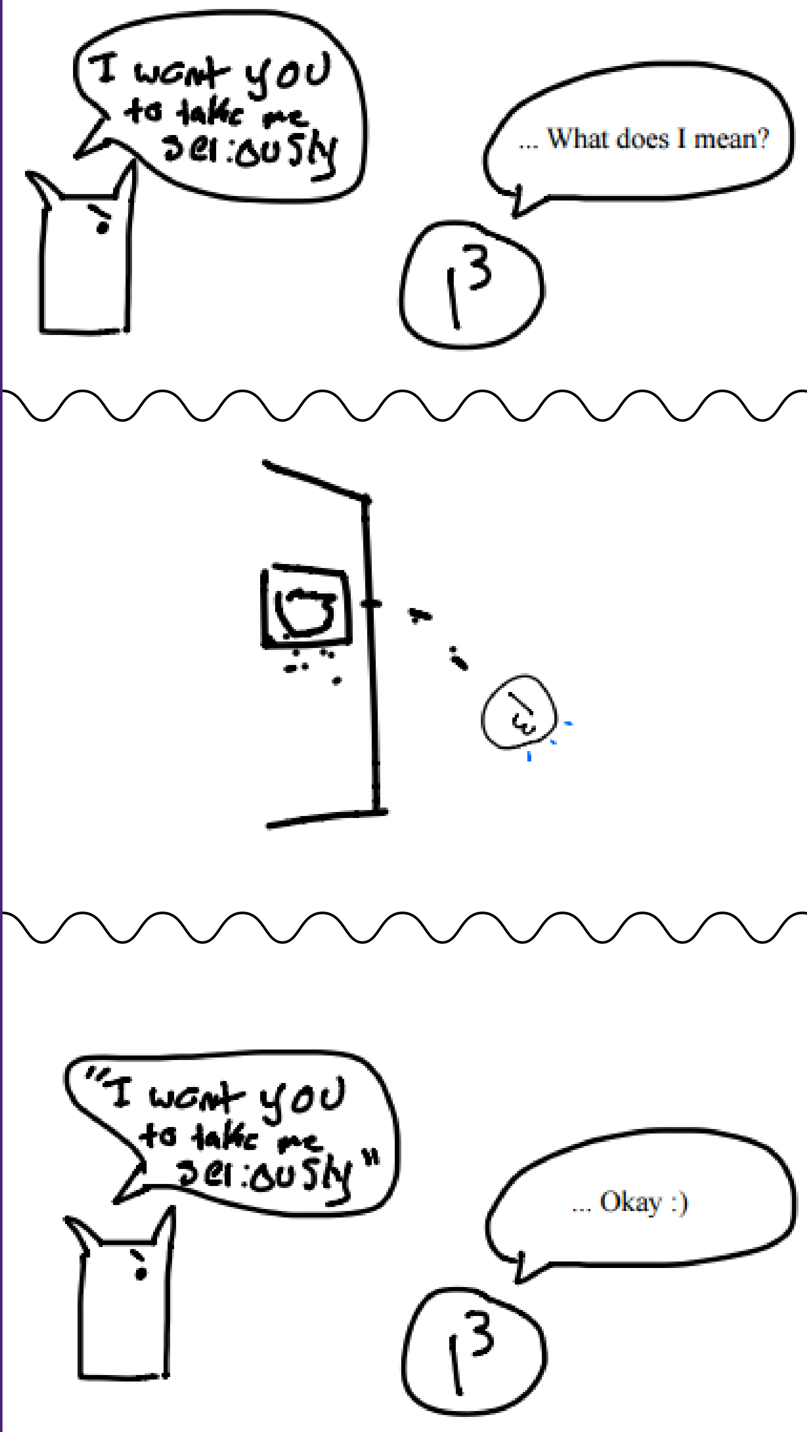
Ignore the gray window and look at the black bar at the bottom

## WHAT WE JUST DID

The whole print() thing is called a function

We first type the command we need, and then put all the data we need the function to work with inside the ()s (and then end it with a semicolon ;) )

When we want the computer to take our letters word for word, we need to put ""s around it or else the computer will think it's a command


I want you to take me serious!y
... What does I mean?
"I want you to take me serious!y"
... Okay :)

Carl
Your number is five
Carl
5
L8R
Carl, what is your number?
5
5
Thanks

# VARIABLES

A variable is a place that can store a number (and a few other things). You can make the variable store a different number later. You are able to name the variable so that you can use it whenever you need easily.

Variable Types We Use A Lot:
Integer (Whole Numbers)
Float (Whole Numbers & Decimals)
Boolean (True or False)
String (Words & Letters)

# HOW TO MAKE A VARIABLE

To make a variable, you first write out the type of variable you want, then the name you want it to be, an equals sign, and then the value:

Examples with the common type:
int carl = 5; (Integer)
float sam = 4.4; (Double)
boolean isCorrect = true; (Boolean)
String someText = "I am some text"; (String)
Variables are named likeThisWay

Carl
Your number is five
Carl
5
L8R
5
5
Carl, what is your number?
Thanks

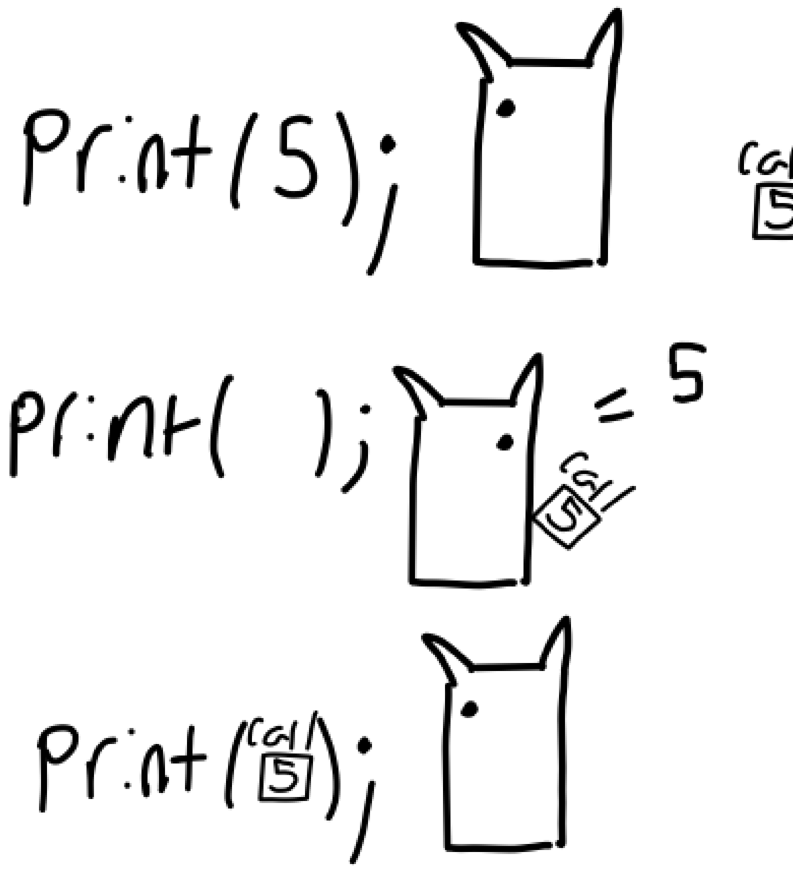
## USING VARIABLES

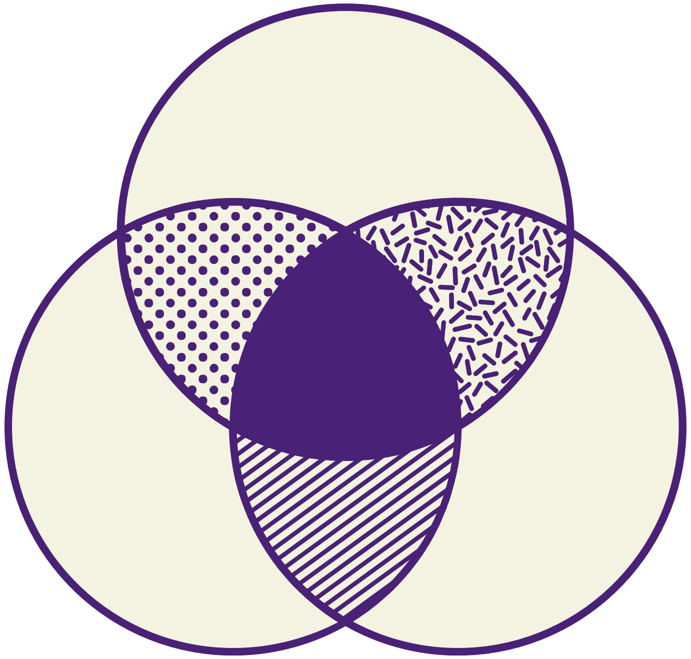
You can treat the variable while coding like you would treat a number, but instead, you type its name.

You can use one inside parenthesis for functions or use one to add:

```
String someText = "Hello World!";
println(someText);
```

This is why we need ""s when we want to type something literally. If it's not, the computer thinks it's a variable.





## CHANGING UP VARIABLES

Variables can be equal to creative things. You can set them equal to another variable, or equal to an equation, or an equation with another variable.

```
int sally = 3;
int carl = sally + 5;
```

Carl is now 8.

After a variable is made, you can change it:

```
carl = 5;
```

Now carl is 5

## TRICKS TO CHANGE VARIABLES

```
int carl = 0;

carl++; (Add one to carl)
carl--;  (Subtract one from carl)
```

You can add a variable to itself like:

```
carl = carl + 2 (Add two to carl)
carl += 2 (Add two to carl)
```

## SHAPES!!!

The way to draw a rectangle or a circle is the same as to print to the console, it just has a new name and different data in the ()s

```
rect(XPos, YPos, length, width);
ellipse(XPos, YPos, length, width);
```

All of the shapes and all of the cool functions in processing can be found at: processing.org/reference

## OTHER USEFUL FUNCTIONS

size(length, height) - Change the size of the window

surface.setResizable(true) - Make the window resizable

fill(number) or fill(r, g, b) - Color for the next objects you make
(Out of 255 for each)

noStroke() - Make shapes not have outlines

# OTHER USEFUL FUNCTIONS

background(number) or background(r,g,b) - Color the background of the program

println(text or number) - Makes a new line for every text printed

Tip :
("Hello " + " world") - The + can combine text in one line by using the plus, it won't add them it will combine them. Include spaces in the text

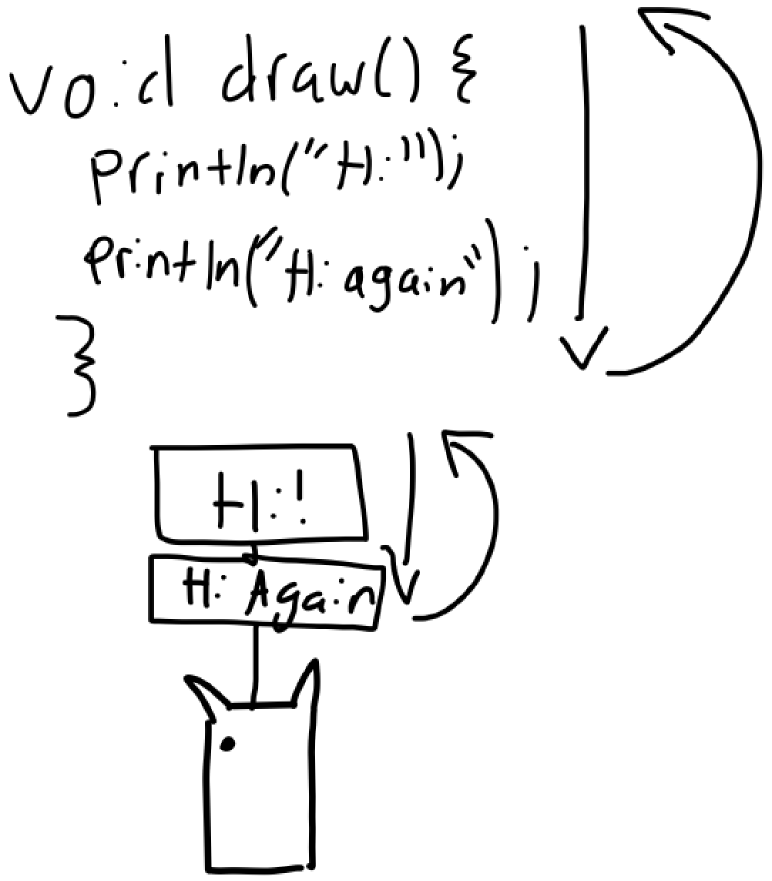
## DRAW AND SETUP

Methods are blocks of code grouped together and labeled by a name

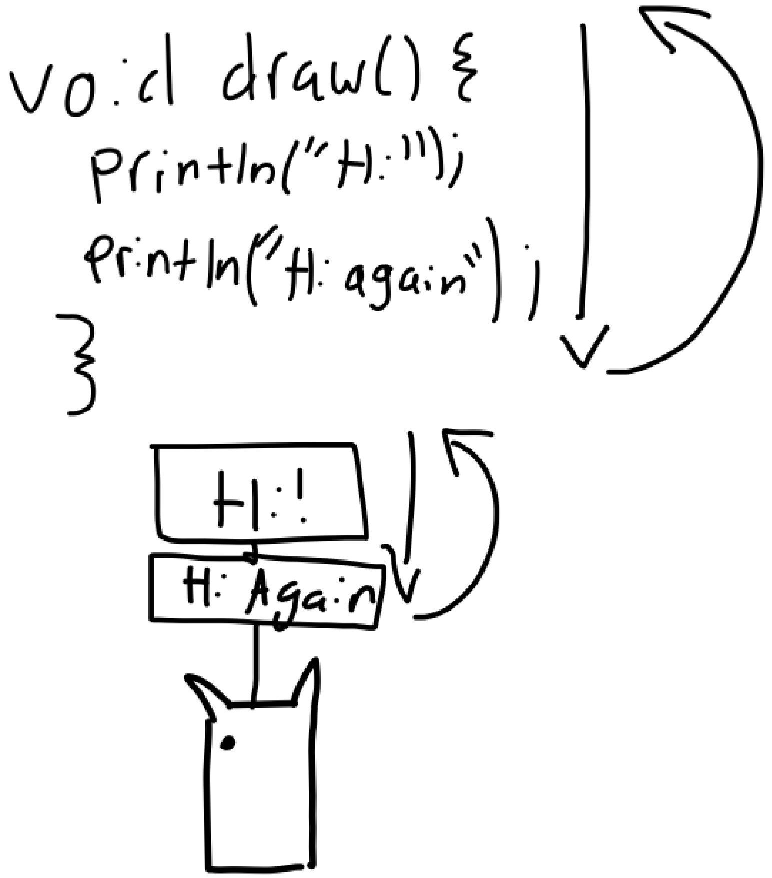
Draw and setup are two methods that you can make that each have a separate purpose

The program will run through all the code in the setup() code block or method one time when the program starts

The program will keep repeating, or loop, through the code in draw() forever



```
void draw() {
  println("Hi!");
  println("Hi again");
}
```



## HOW TO USE DRAW AND SETUP

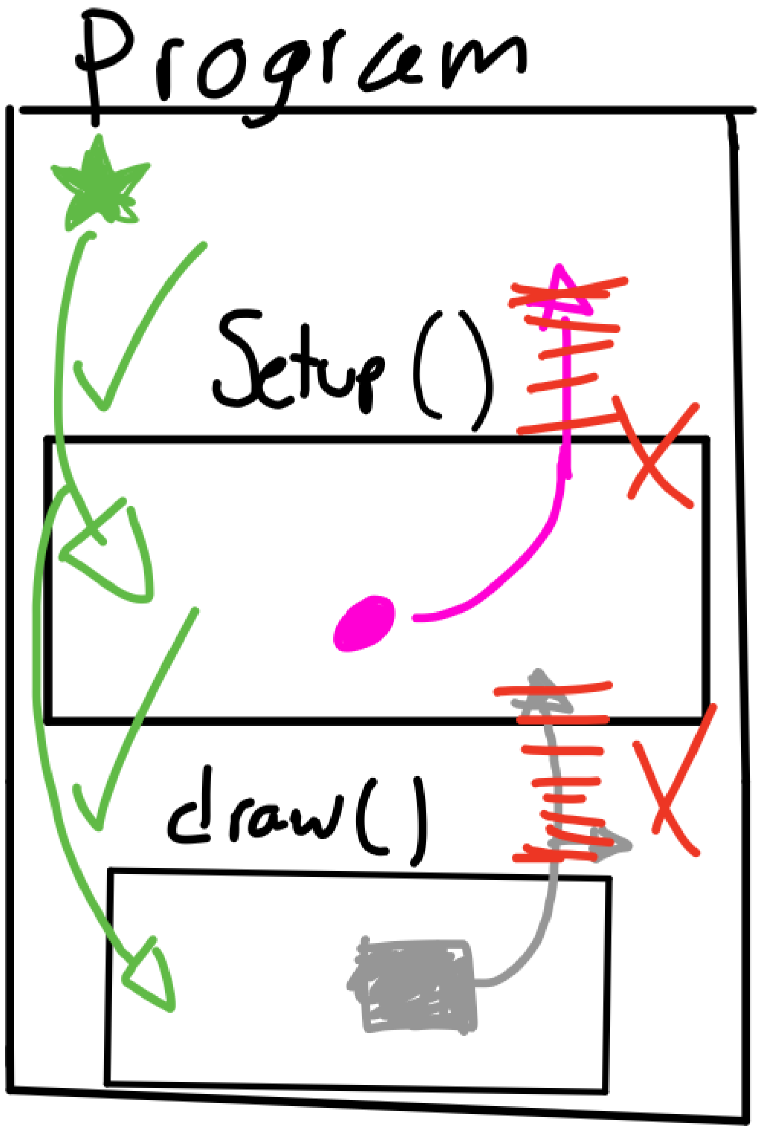
```
void setup(){
 size(800, 800);
 println("Hello for the first time!");
}

void draw(){
 println("HELLO AGAIN! HELLO!");
}
```

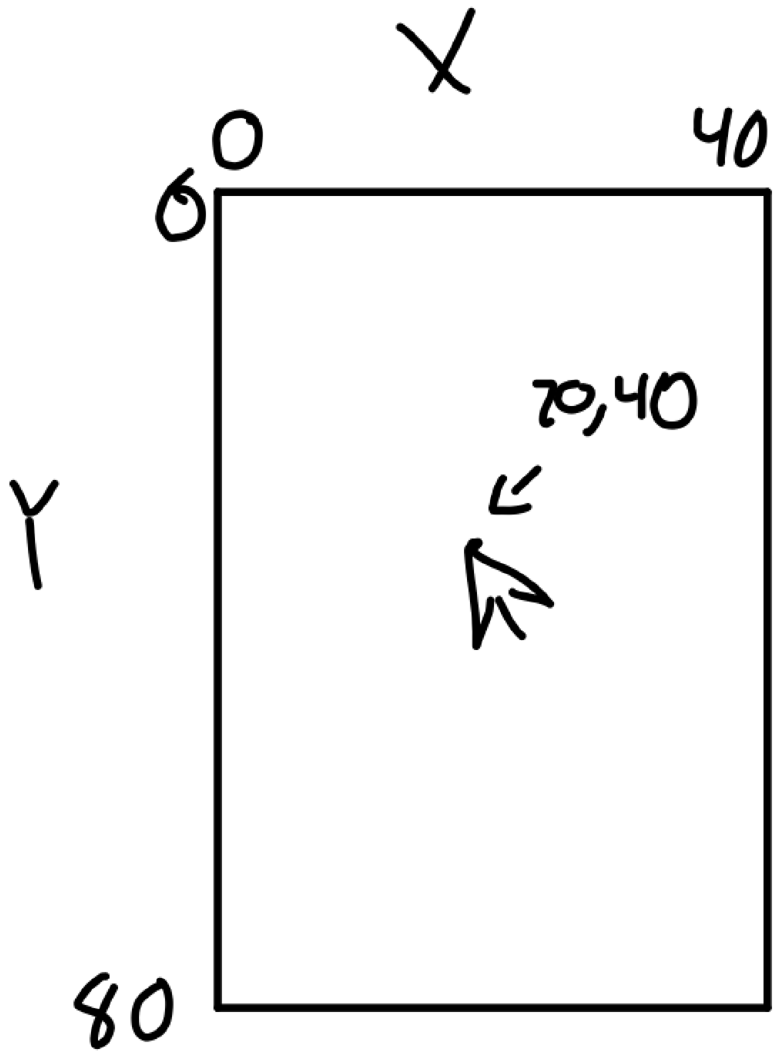
All your code goes inside the brackets.
Don't worry about void

## VARIABLES WITH METHODS

If a variable is made inside of a method (again, a code block like draw()), then it can only be used inside that method. This is called a local variable.

If a variable is made outside of all methods, (like at the top of the program) then it can be used and changed by all methods. These are called global variables.





## A LITTLE BIT MORE ON VARIABLES

There are some VERY useful variables that are already made by default (you don't have to make them, they just exist and you can use them):

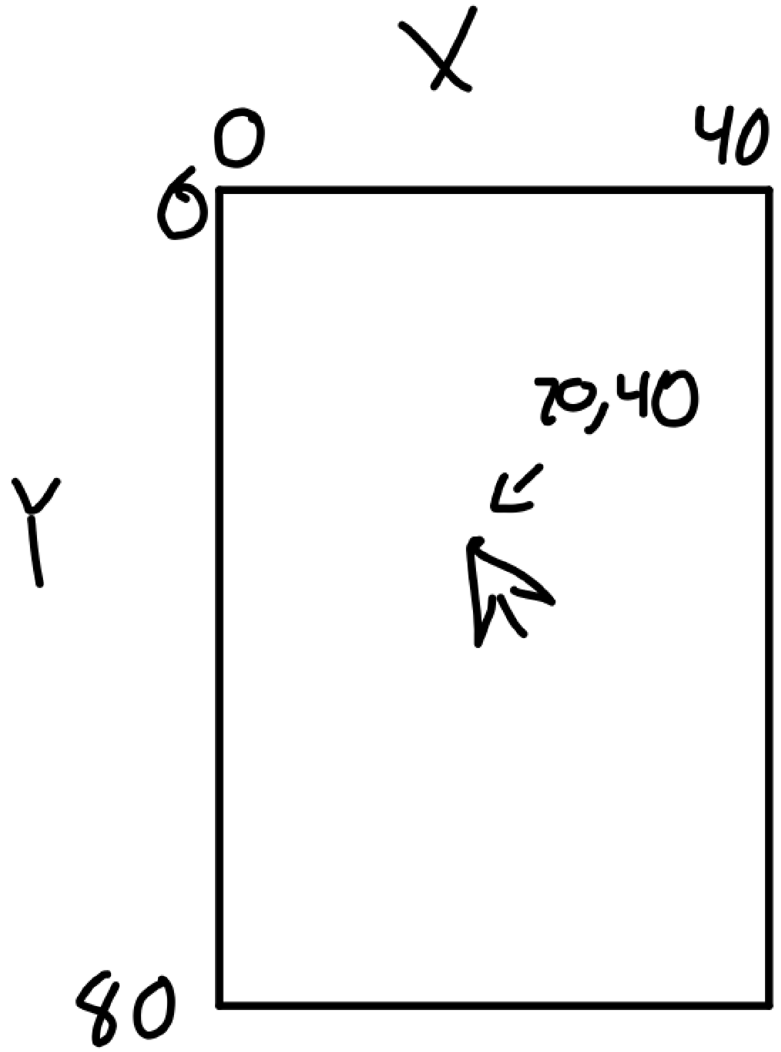
width & height - The length and width of the program's window

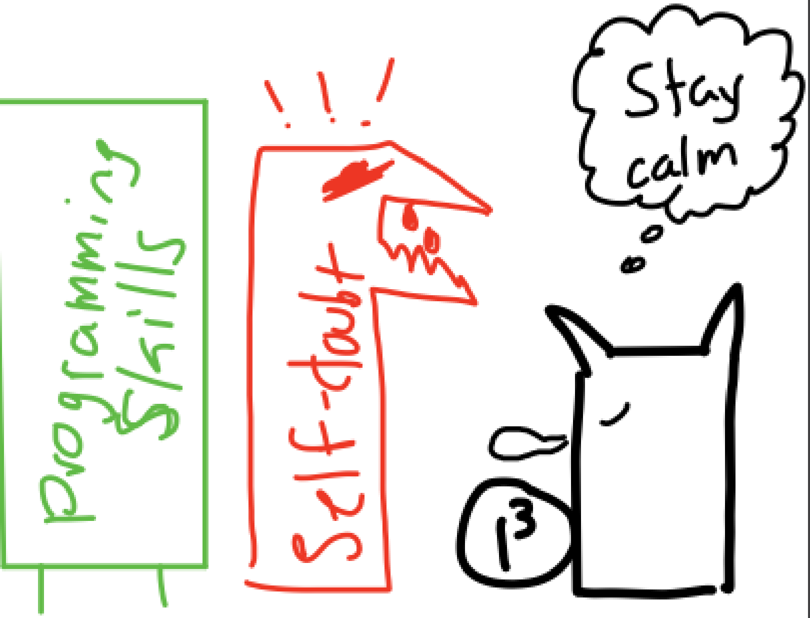
mouseX & mouseY - The current position of the mouse on the program's coordinate grid

## COORDINATE SYSTEM

The coordinate system looks like the picture on the right. There are no negatives that appear on the screen, so if you set it a rectangle to a negative x or y value some of it will be off the screen.

The X and Y start in the top left
The X increases as you go right
The Y increases as you go down





## FIRST ANIMATION

We want to make a circle go from the left side of the screen to the right. We can do this with all the current information, and I'll make it easier with a few pointers

I'll have a few questions to help, and then we'll do it together

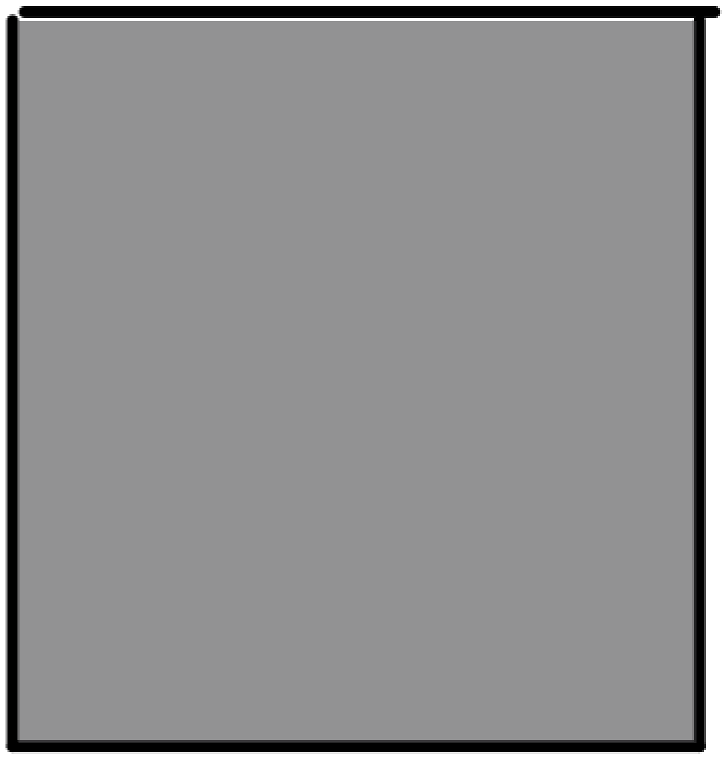
It's okay if you don't get it or don't know where to start. The more we go over this stuff they better you get at figuring how to do it. Stay calm

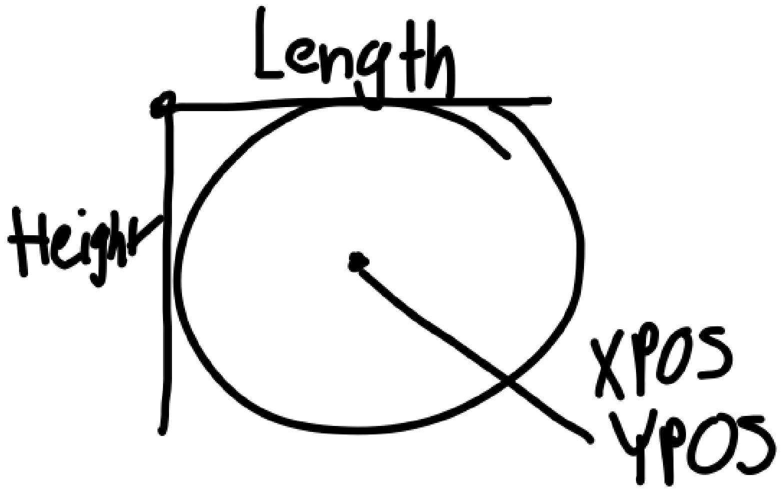
## PT 1 - SETUP THE PROGRAM

Let's first make the base for the program. Start with our base setup:

```
void setup(){
size(800, 800);
background(100);
}

void draw(){

}
```

This will make a gray 800x800 window





## PT 2 - MAKING THE CIRCLE

The proper way to write a circle, or the syntax, is:

```
ellipse(XPos, YPos, length, width);
```

Since we want a circle, we'll want the same length and width, so let's make it 100x100

```
ellipse(XPos, YPos, 100, 100);
```

Length starts on the left side, not the center of the circle

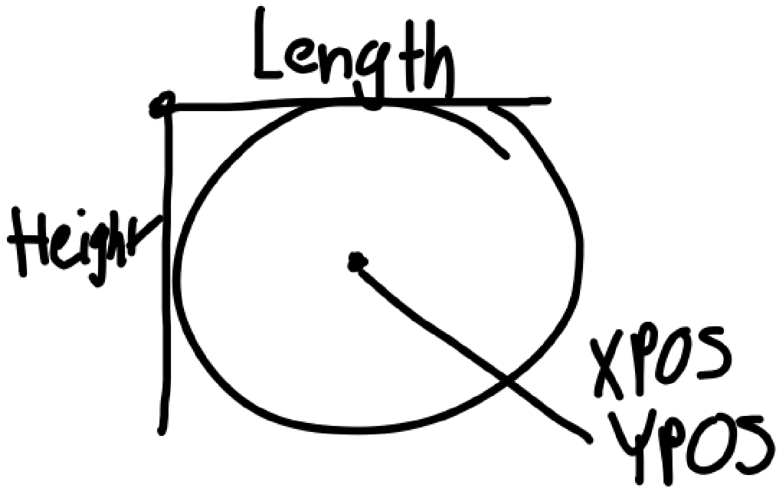
## PT 2 - MAKING THE CIRCLE II

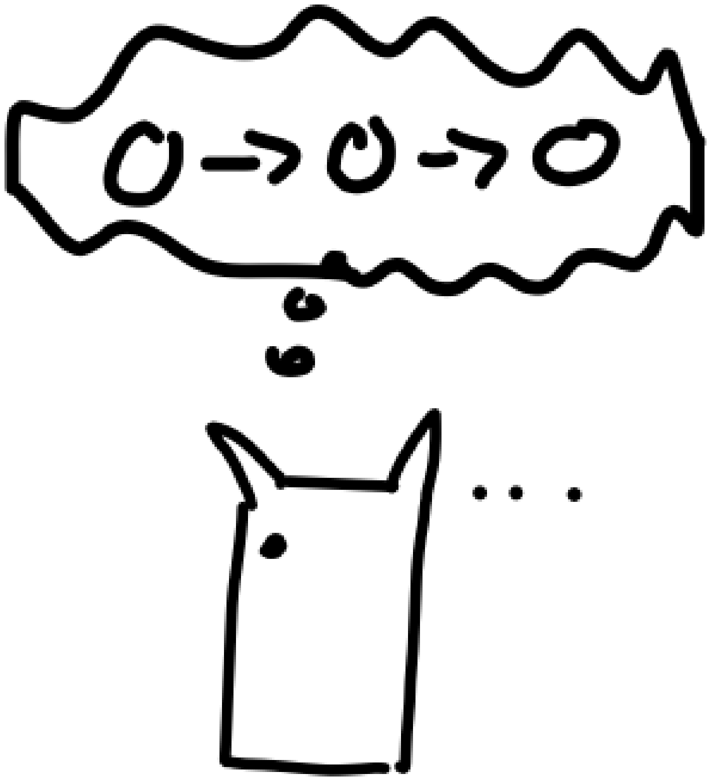
```
ellipse(XPos, YPos, 100, 100);
```

Now for the coordinates, the X and Y are the center pixel of the circle. Let's put it in the middle of the left side of the screen by taking the height of the program (800) and dividing by 2 (400), and having X equal 0.

```
ellipse(0, 400, 100, 100);
```

ellipse(0, height/2, 100, 100); would work also





## PT 3 - PROBLEM SOLVING

Now we have a circle that doesn't move. There is no function we know that will move it to the right, so we have to think more creatively. What can we do to make the illusion of movement based on the ellipse's data we have to put in (parameters)?

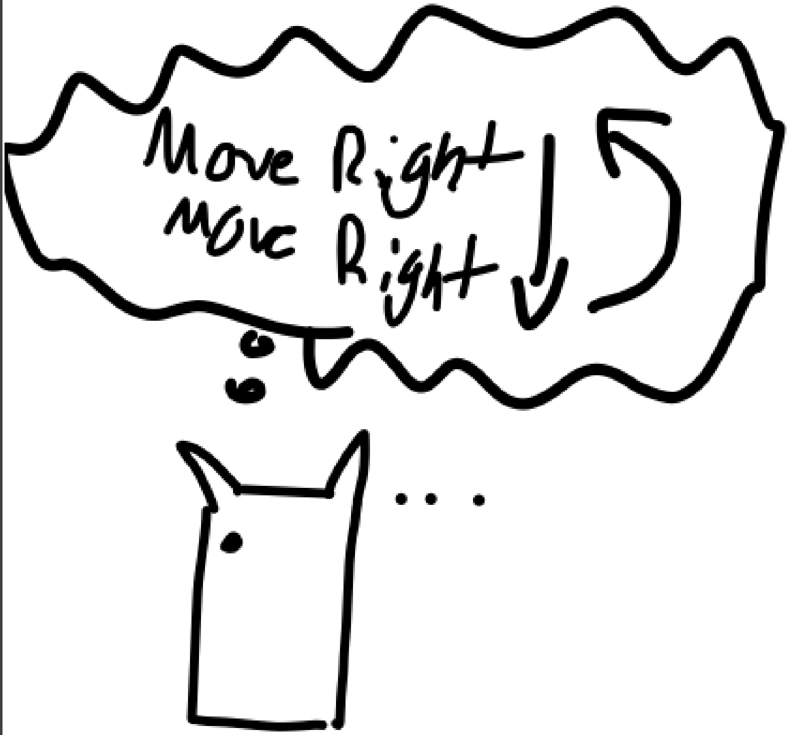
HINT:

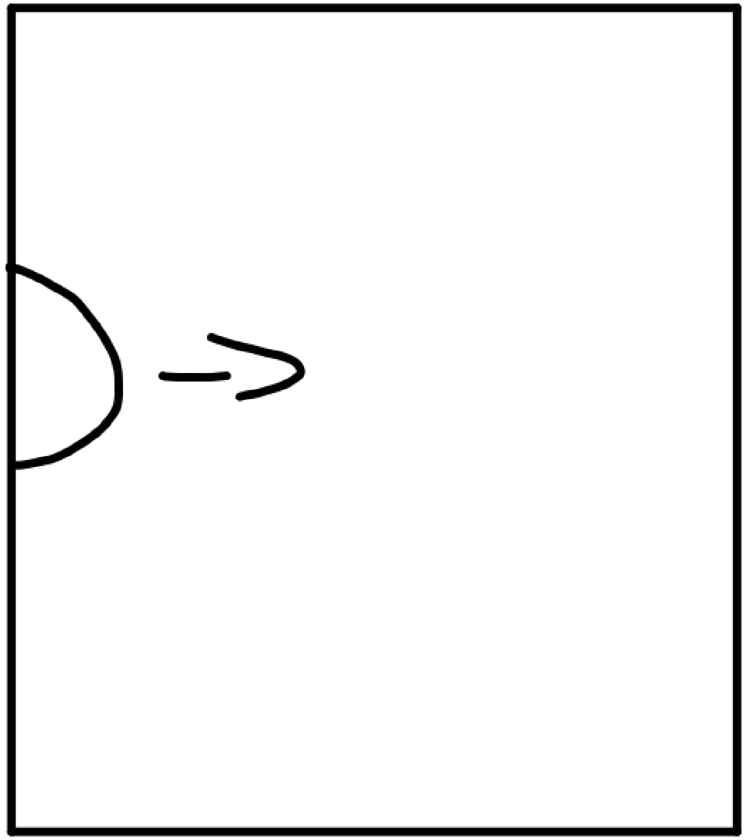
What happens on a coordinate grid when something goes the right?

## PT 3 - PROBLEM SOLVING II

Now what could we do in order to make the X position of the circle keep going right?

Hint:





## PT 4 - MAKING THE VARIABLE

Let's make our variable xPos. We need to have the variable increase by 1 constantly, which we can do using the draw() method. So our program will be -

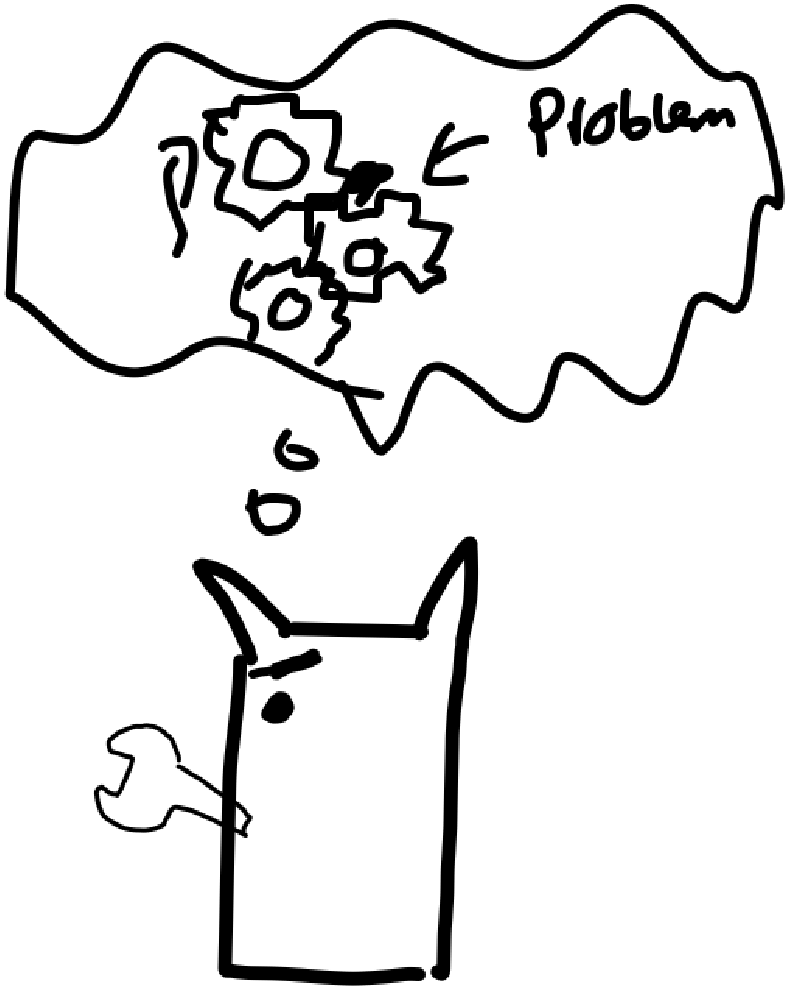
```
void setup(){
size(800, 800);
background(100);
}

void draw(){
int xPos = 0;
xPos++;
ellipse(xPos, 400, 100, 100);
}
```

## PT 5 - TROUBLESHOOTING

Why does the circle not move?

Hint:

Look at where xPos is being made and what we're telling the computer program



## PT 5 - TROUBLESHOOTING

Why is that streak happening? Oh wait, did we even make the circle move? What did we really do when we used the ellipse() function?

What can we do to fix this?

Hint:
If we make a new circle every time, how can we go over the previous one?

## PT 6 - GET MORE CREATIVE

What other things would you like to play around with to make this more interesting? What would you do to make it look better? How could you change how the circle moves?

As we learn how to make a computer do something if something happen, we can start to become even more creative

## PROCESSING INTRODUCTION