PROCESSING LOGIC

LOGIC IDEAS

Logic in programming usually deals with checking if a statement is true or false:

if (something is true) -> (then do something)

When we want to check if something is true, we use an **if** statement. If it is not true then the program skips over the "do something" part. You don't have to put "is true", the if statement checks for it itself.

IF STATEMENT

```
if (something)[is true] {
   do the stuff in here
}
```

An example :

```
if (5 > 4) {
  rect(40, 40, 40, 40);
}
```

LOGIC OPERATORS

Logic operators are the signs used to compare things to see if something is true:

== - Is equal to
> - Is greater than
>= - Is greater than or equal to
< - Is less than
<= - Is less than or equal to

LOGIC EXAMPLES

```
x = 6;
y = 6;

if (x > y)
  println("X is greater than Y");

if (x == y)
  println("X is equal to Y");
```

This would print out :
X is equal to Y

If only one thing needs to be done, you don't need the {}s

BOOLEAN VARIABLES

Boolean variables are variables that can either be true or false.

```
boolean isCorrect = false;
```

Like with mouseX and mouseY, Processing makes some boolean variables automatically. The really cool ones here are mousePressed and keyPressed, where if the mouse or key is pressed, then it will equal true, and if it isn't, it will equal false

BOOLEAN VARIABLES

Variables can be put into an if statement by itself:

```
if (mousePressed)
 background(4);
```

The program will ask "Is mousePressed true?", and if it is, then it will activate the background command. For most cases, we want these to be in the draw() method, so that it constantly check if the mouse is pressed instead of just when the program starts

ELSE STATEMENTS

If the thing inside an **if** statement is false but an **else** statement is below it, then the program will do whatever is in the **else** statement. An **else** statement will activate if the **if** is wrong

If (something) [is true]
(This will be done)
else [if the something is false]
(This will be done instead)

ELSE EXAMPLE

```
x = (a number);
y = (a number);

if (x > y)
  println("X is greater than Y");
else
  println("X is NOT greater than Y");
```

Why can't the **else** statement say "X is less than Y"?

ELSE IF

An **else if** goes between and **if** and an **else** statement. An **if** statement needs to be above it for it to work, but an **else** statement doesn't have to be below it.

The easiest way to think about it is
if - "if something is true do this and skip over the rest"
else if - "else if something else is true then do this"
else - "else if none of the statements have been true, do this"

ELSE IF EXAMPLE

```
x = _;
y = _;

if (x > y)
 println("X is greater than Y");
else if (x == y)
 println("X is equal to Y");
else
 println("X is less than Y");
```

You can put as many else if statements as you want

NOT TRUE

! in programming means the opposite of the true or false of something.

!true == false
!false == true

This is used to flip what you would normally expect, and can be put in front of a group of something like !(x > width). It will follow PEMDAS.

NOT TRUE EXAMPLE

(Put inside the draw() method)

```
background(255);
if (mousePressed)
  background(0);
```

See the difference when compared to

```
background(255);
if (!mousePressed)
 background(0);
```

AND / OR SYMBOLS

AND == &&
OR == ||

```
if (something && another thing)
 do something
if (something || another thing)
 do something
```

If you put an AND symbol inside an if's ()s, then both logic statements need to be true for the program to do something. If you put an OR symbol, then either, or both, can be true for the program to do something.

AND / OR EXAMPLE

Do this if statement first:

```
background(0);
if (mousePressed && keyPressed)
 background(255);
```

Both the mouse and a key have to be pressed for the screen to turn white. Now replace the && with an || and try it out. Now either, or both, can be true for it to work!

IMPROVE OUR ANIMATION

We last week (or maybe today!) learned how to make a circle slide from the left side of the screen to the right. However, once we got to the edge, it went off into the unknown. Wouldn't it be more fun for it to bounce back between the walls forever? Well ... now we can! It may just not be apparent, but we have all the concepts to make this happen, now you just need the experience!

PT 1 - ESTABLISH GOAL/IDEA

Let's first see in a word sense what we're going to add. We want to say **if** the center of the circle (we can change it to the side later) goes past either side of the screen, we want it to reverse directions ... how could we say the first line of the if statement using one of the circle's variables?

PT 2 - MAKING THE IF STATEMENT

We want to add our if to the (draw() / setup()) method (Circle one!)

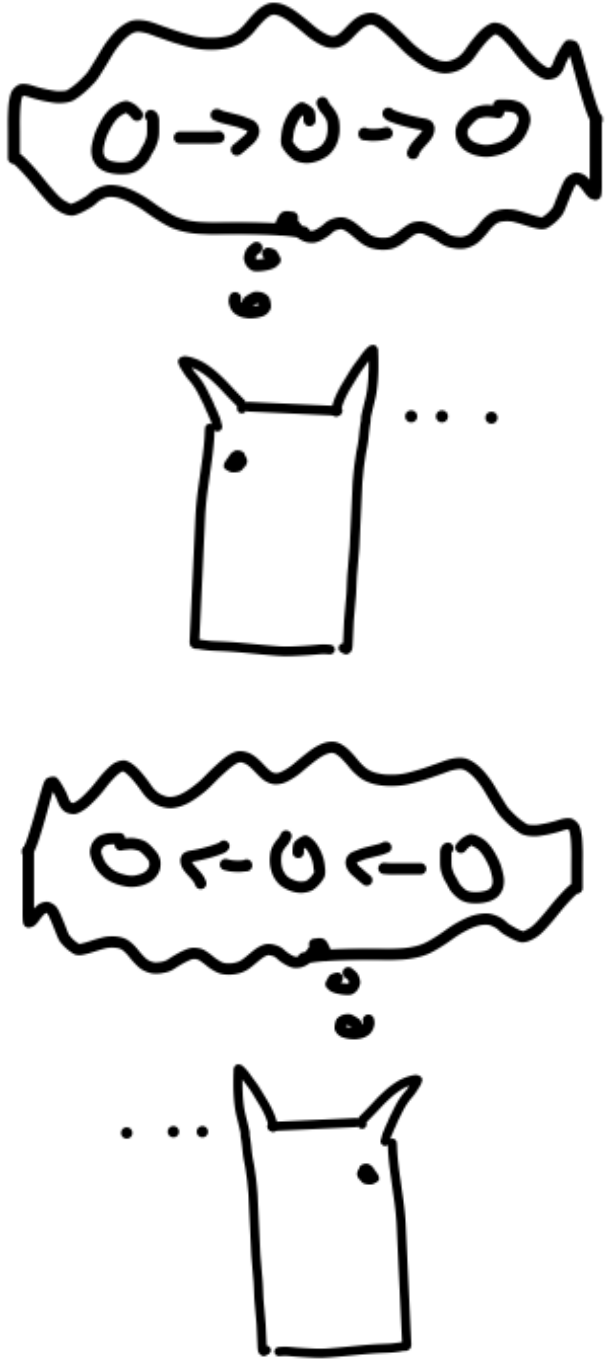
This is what it looks like:

```
if (xPos > width || xPos < 0)
  (reverse circle's direction)
```

This is saying "**If** the X of the circle is less than **or** greater than the window size, reverse its direction so the circle is going back into the window

PT 3 - REVERSING THE DIRECTION

Now we need to make the circle head in the other direction until we hit that side, then back again, then back again forever. Essentially, whenever the ball hits the wall, we want it add a number that will go in the opposite direction of where it's currently going. The best way to illustrate how we'll do this is with a math example I'll show now.



PT 3 - REVERSING THE DIRECTION

So we need to add a number to the xPos the opposite of what is currently being added. Can we just say if the xPos is greater than the width we just subtract 10? Why?

What can we use that can **change**, and can be **added** to the xPos **depending** on where the xPos is?

PT 4 - MAKING THE VARIABLE

Let's make a new variable let's say speed, directly below our xPos coordinate.

```
int speed = 10;
```

Now, using what we covered, how can we say that if the xPos is greater than or less than the sides, we reverse the directions of speed.

PT 5 - THE FINAL PART

We can make the if statement like this:

```
if (xPos < 0 || xPos > width)
 speed *= -1;
```

Now insert this in the draw function, and see what happens

PROCESSING LOGIC