If() or Else

```
if(X>2)
{
//do something
}
else
{
//do something else
}
```

Making something react to a variable is great! But what about when you want something to happen at a certain value, or if one value is higher than another? Enter the if() statement.

PROCESSING SKILL REQUIREMENTS
DIFFICULTY 1-5

ELECTRICAL PROTOTYPING
ROBOTICS
SOLDERING
PROGRAMMING
DIY

MATERIALS LIST

- Computer
- Processing

STEP 1: Draw Something in Processing

To start, draw a basic geometry in the center of your window. We are going to be using this for a number of different uses of the if() statement. Once you have mastered how to to use if() and else statements, your coding skills are going to blossom. When you master one shape for simplicity, the project will progress to you hacking your holiday image code!

STEP 2: Variables and Data Types

We are going to be using a rectangle as an example. Up to this point we have either been using static numbers, or what are called "system" or built-in variables (`mouseX`, etc.). We are now going to create our own variables. Variables store information such as numbers, letters, etc. Each variable has a data type that describes that variable. Here is a list common variable data types:

`int`- integer: a whole number (e.g. **12**)
`float`- float: a number with a decimal (e.g. **2.1**)
`char`- character: letters or numbers bound by apostrophes (e.g. **'a'**)
`String`- string: a series of characters bound by **"quotes"**

```
int X= 50;
int Y= 50;

void setup()
{

}

void draw()
{

}
```

We normally name and set our variables at the very top of a sketch, even before our `setup()` code. We are going to create two variables - X and Y - for each of our rectangle's X and Y coordinates, and they will both equal 50. This is called initiating a variable. We are doing this so that we can change them freely during the sketch later on. Notice that we still end each line with a semicolon!

STEP 3: Assigning Variables to Functions

We can now use X and Y in our `rect()` function; note they are currently set to equal 50 at the beginning of the sketch. The width and height are constants of 100, but we can now change the X and Y to whatever we want with some basic operations.

```
int X= 50;
int Y= 50;

void setup()
{
  size(200,200);

}

void draw()
{
  background(150);
  rect(X,Y,100,100);
  X++;
}
```

STEP 4: To Increment or Decrement

If we wanted to increase the X over time, or make it move right and left, we can increment or decrement X by using this line of code: X= X+1;. Or, we can just type X++; – both increment X. How would you decrement X? You can do the same thing to Y as well. See the example code for incrementing X. Try it out and play for a bit (note: you may want to make your `size()` larger.)

If you wanted to change X or Y by a number larger than one, it would look something like this:
X+=5; (increment by 5)
X-=10; (decrement by 10)

STEP 5: Keeping it in Your Sights

It's cool that we can make a shape move, but we want to keep it in the sketch window. This is where an `if()` statement is handy! We want the rectangle to move but stay in the window. So, we are going test if X is larger than the width of the window using an `if()`. The basic structure of an `if()` statement is shown here. For example, we can create a statement like `X>width` and add it to an `if()` statement. If that is true, we want to multiply a new variable called grow by (-1). We created grow as a variable so that we can change it negative or positive depending on the direction the rectangle is going. See the example code for what we are looking for.

STEP 6: OR

The last step is to keep the rectangle bouncing off *both* sides of the window. As it stands now, it only bounces off the right side. We can change the mathematical statement to include OR (||), which will allow us to say, "if X is larger than `width` OR smaller than 0," bounce! The way we type OR is ||, which is the pipes key (SHIFT + \). A couple other operators are:

&&- AND
||- OR
!- NOT

Check out this last piece of code for creating a bouncing square. We have not only added the OR, but also subtracted the width (100) of the rectangle from the `width` of the screen, so it bounces off of the right edge of the rectangle rather than the origin. Now it's your turn! Use an `if()` statement to move a shape, change color, or even make it appear/ disappear altogether.

```
int X= 50;
int Y= 50;
int grow= 1;
void setup()
{
  size(200,200);
}

void draw()
{
  background(150);
  rect(X,Y,100,100);
  X= X+grow;

  if( X>width-100 || X<0)
  {
    grow=grow*(-1);
  }
}
```

You can also add an `else` statement after an `if()`. The else happens if the mathematical statement is false. So a basic structure looks like:

```
if(mathematical statement)
{
    // do something if true
}
else
{
    //do something if false
}
```

STEP 8: Share your work!

If you would like to share your work - which you do - you can sign up at openprocessing.org. Open Processing allows you to share your code and view the drawing online, and allows others to fork or borrow your code to manipulate and change it on their own, while still keeping your sketch intact for others to check out.

OpenProcessing

TAKING IT FURTHER

- Try using math and variables to make interaction more intricate.
- Make a snowman run around in your holiday image.