Lesson 11: End Conditions

Lesson aim:

To build on our knowledge of variables and "if" condition to make cool animations!

Why:

Every game needs end conditions as well as advanced animations that we will use in our project. Looping shapes (so they never stop and loop from one side of the screen to the other) will be used in our project.

Recap: "if" statements

If what is between "(" and ")" is **true**, execute the code in the following ***{code block}*** (that is the code between "**{**" and "**}**"). If not, don't execute that code - just move to the next instruction after the ***{code block}***.

We also looked at the following keywords, to help us work with user input:

`if(mousePressed == true){ /*execute these instructions*/}`	if the mouse is pressed, execute instructions in the ***{code block}***
`if(mouseButton == LEFT){ /*execute these instructions*/}`	If left mouse button pressed, execute instructions in the ***{code block}***
`if(keyPressed == true){ /*execute these instructions*/}`	If any key pressed, execute instructions in the ***{code block}***
`if(key == 's'){ /*execute these instructions*/}`	If 's' key pressed, execute instructions in the ***{code block}***

Test your knowledge!

- Write a program that changes the colour of a circle depending on whether the 'q' key is pressed.

We can also do other comparisons, have a look at the following example (let it run for a while):

```
int ballHozPosition;  //← make new variable and name it
void setup(){
 size(600,600);
 ballHozPosition = 50; //give variable starting value
}

void draw(){
  ellipse(ballHozPosition,300,100,100);
  if(ballHozPosition < 200){ //if ballHozPosition less than 200
    ballHozPosition = ballHozPosition + 5; //update position
  }
}
```

- Create a new program and try out the code above. What does it do?

Conditions! With more conditions comes more power!

Fill in the following table with the meanings of the condition symbols of the left:

Condition	Meaning
==	is equals to
>	
<	
>=	is greater than or equal to
<=	

Tasks:

1. **Fix the example!** Stop the ball when it gets to:
    a. The middle of the program screen. What number do we need to change? To what?
    b. The right edge of the screen.
        i. Modify this so the ball stops when right side of the ball hits the right edge of the screen - not when the middle of the ball hits the edge. I think this looks best!
2. Slow the movement of the ball in the example - look at code from lesson 10 if you're not sure how.
    a. Now speed its up! How fast can you make the ball go and still see it?
3. **Looping**! We have stopped our ball, but that gets boring, the ball stops moving! We don't want that! Make a new program that moves a ball from left to right. When the ball reaches the right hand side of the screen i**t should disappear, and reappear on the leftmost part of the screen!** This is tricky. We need to change our condition. Rather than move the ball until it gets to a point (which is what we have been doing), test when it gets to the edge and **then** change the position (stored in the variable) to the edge of the left side of the screen. Ask an instructor if you need a bit of help.
    a. The ball and background should be coloured and you should only see one ball at a time.
    b. Make the animation smoooooooth. Only change the position when the left side of the ball leaves the right edge of the screen, and then only the rightmost side of ball should appear!