What will we be learning in this lesson?

- We will go into variables in more detail.
- Start to do some basic animations.
- Talk about the power of the code blocks a bit more.

Variables:

We were introduced to variables in the last lesson. When we use the term variables, we're referring to **data that can be altered while the program is running**. Using variables is a way to store this data in our program. Using variables allows us to create more complicated programs, like games, animations and other applications.

In order to create and use a variable in our program, we need to follow three basic steps. You must declare it, give it a starting value and then update the variable while your program is running. There a lots of different types of variables, each one deals with different types of information. Today, we're just gonna use one type of variable, a `float` variable.

What are the compounds of a variable:

Take a look at the following variable: `float aNumber = 10;`

The `float` as we have discussed is the type of variable. In this case it's a number. `aNumber` is the name of the variable. This is how we call a variable. Two variables cannot have the same name. `10` is the value that is currently assigned to the variable, which depends on the type of the variable.

Floats - What they are and how to create them:

`float` is short for "floating point number". A `float` can be any number between minus three million to positive three million. `float` variables are very useful in coding. Using just floats can allow us to do some cool things. We're gonna write our first program using `float` variables now, which will also be our first animation.

NB: We recommend you open a new processing file and name it Lesson3FirstAnimation

Task:

We're gonna go through a step by step instruction on how to create this amination. If your really stuck, ask a teacher for help.

1. On the first line of your program write: `float ballHozPosition;`
   This **declares** (creates) the variable `ballHozPosition` of type `float`.

2. Inside `void setup()`, set the size of the screen and write `ballHozPosition = 50;`
   This is how we **initialize** (give a value to) our variable.

3. Inside `void draw()`, Create a ellipse with the horizontal position `ballHozPosition` and write `ballHozPosition += 1;` This is how we update our variable.

If you've done the steps correctly, you should see a circle moving from left to right across the screen.

We have just created our first animation, now we can mess around with it and add more.

Task:

1. Change the colour of the ball.

2. Change the speed of the ball. ***Hint****: Think about how we are updating the variable.*

3. Change the program so that we can only see one ball at a time. ***Hint****: use* `background()`.

4. Make the circle start on the right and move to the left. ***Hint****: You need to change where the circle starts and how you update it.*

Bonus task:

Change from a circle to a square. You can use the `rect()` function to do this. See if you can figure out the differences between `rect()` and `ellipse()`.

How does the animation work?:

What is happening in this program, is that we are drawing multiple circles every second. The reason that we are using `void draw()` in this program is that it runs **60 times a second**. So any code that we have inside this block is run 60 times a second. Once the end of the block has been reached, it loops back up to the top of the block.

In our program, we add one to our variable each time we loop through the code. In the next loop, a new circle will be drawn with this updated position. This happens again and again, making it look like the circle is moving across the screen.

We're now gonna try some more complicated animations.

NB: We recommend you open a new processing file and name it Lesson3Animation.

Task:

1. Create a program that will move a ball from the top of the screen to the bottom of the screen, using a `float` called **`ballVertPos`**. ***Hint****: think about which parameter of* `ellipse()` *determines its vertical position.*

2. Add a second `float` **`ballHozPos,`** and have the circle go diagonally down. ***Hint****: You'll need to update both variables.*

3. Adapt the program, so the circle starts in the bottom right hand corner and goes up and to the left.

Bonus task:

Create four balls that start in the centre of the screen and move directly away from each other

Questions for the end of the lesson?

- How many times a second does `void draw()` run?
- What is the full name of a `float`