Once you start writing code, you will find out that parts of the code repeat themselves. This is normal in programming. When you happen to find this kind of pattern, you can make it reusable by creating a **function**.

A function is a separate piece of code that can be called on demand, by specifying its name in the execution. Here is an example of add(x, y).

Good to take note here: Function and method are used interchangeably in most cases, but there is a difference: functions can be stand alone while methods are bound to a specific class. But what is a class?

What happens when we need a different data type combined with specific functionality. Let's say a player that has a function to take damage. In this case, we can use a **class**.

A class can be defined by simply the class instruction and a name on top of any script.

We can make use of a class in another script. Another advantage of classes is that they can be inherited. That means that all their variables and functions trickle down to their children while adding more functionality.

Let's take the simple example of a base enemy which has health and take damage. Then, we can proceed and inherit from that to create a soldier, a big boss and whatever enemy type we need.

Now that we know what a function is, let's take it one step more: enter recursive functions. When a function has calls to itself it's called a recursive function.

But having calls to itself might lead to infinite loops! This is a common issue when implementing recursive functions. Luckily, there are ways to mitigate this. One of them is using what is known as "exit conditions". They provide a way to exit the recursive loop.

Let's look at this small example of a function that prints 5 numbers.

That concludes the programming primer! For more about GDScript, I've linked some great resources below.