Currying

In this lesson you will be introduced to a method of writing functions with multiple parameter lists.

Currying is a style of defining functions where essentially every function is mapped to an expression that consists of *nested anonymous functions*, that in turn take one parameter each.

In other words, rather than a single parameter list, a currying function is passed or applied to multiple parameter lists.

The best way to understand currying is by looking at a simple example. Let's define a simple summation function which adds two integers together.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def simpleSum(x: Int, y:Int) = x + y

println(simpleSum(3,2))
```

The function above is applied to a single parameter list which contains two parameters.

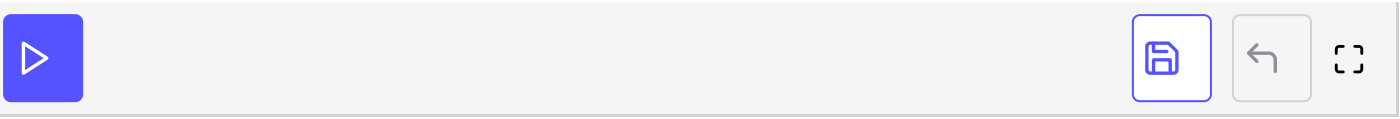
Now, let's define a curried function which has the same functionality as `simpleSum`.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def curriedSum(x: Int)(y: Int) = x + y

println(curriedSum(3)(2))
```



This function is applied to two-parameter lists with a single parameter in each list.

What's happening here is that when we call `curriedSum`, we actually get two back-to-back traditional function calls. The first function call takes a single parameter of type `Int` and returns a function value which will be used by the second function. The function returned by the first function is the second function. The second function, in turn, takes a parameter `y` of type `Int`.

Currying will become a lot more clear in the next few lessons.

In the next lesson, we will be applying the currying syntax to our summation program.