What are Higher-Order Functions?

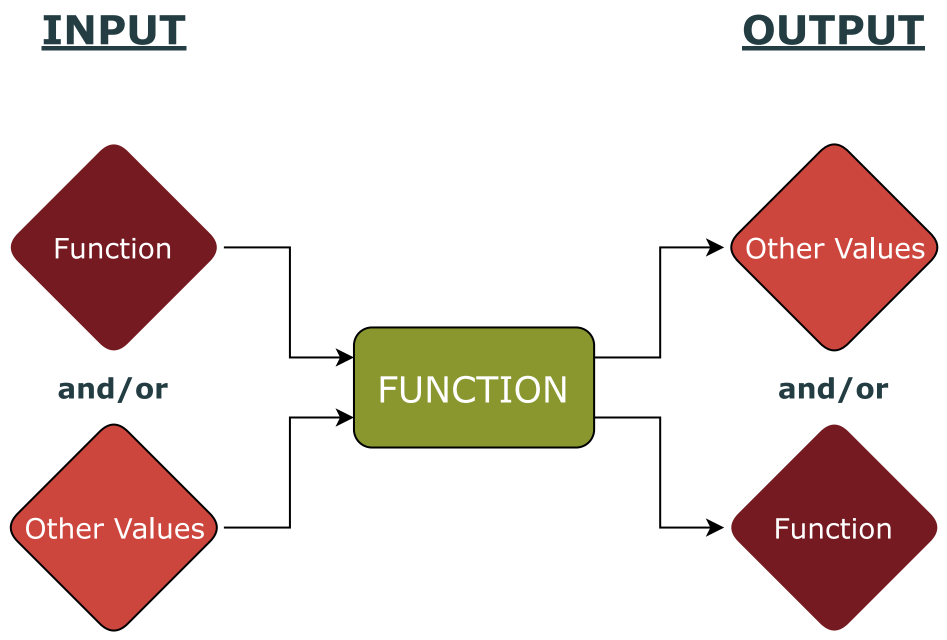
In this lesson, you will be introduced to higher-order functions and learn their syntax.

We'll cover the following

In this chapter, we will see the actual power of functional programming and how functions in a functional programming language differ from functions in a non-functional programming language.

Higher-Order Functions

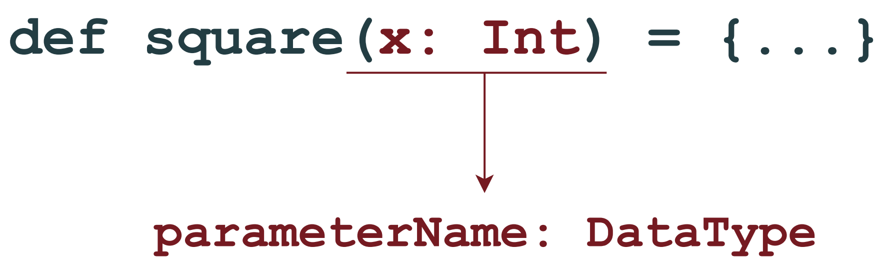
Functional programming languages treat functions as *first-class* values. What this means is that, like any other value, such as Integers or Strings, a function can be passed as a parameter to another function and can also be returned as a result.



Functions that take other functions as parameters or that return functions as results are called **higher-order functions**.

Syntax

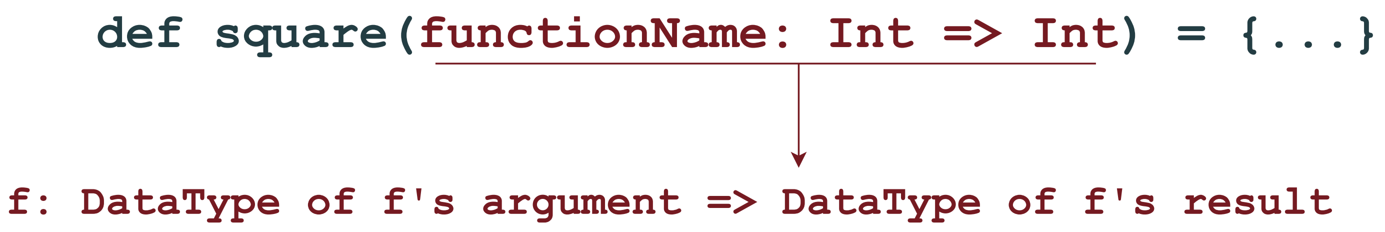
As we already know, the syntax for defining a function which takes a simple variable as a parameter is:



```
def square(x: Int) = {...}

parameterName: DataType
```

When we want to define a function which takes another function as a parameter, the syntax of the parameter would be the following:



```
def square(functionName: Int => Int) = {...}

f: DataType of f's argument => DataType of f's result
```

We have taken the function name to be `f`. `f` is the function to be passed as a parameter. It is followed by a `:` which is further followed by the data type of `f`'s argument. We then insert an arrow `=>` which is followed by the data type of `f`'s result.

In the next lesson, we will be looking at an example which will be carried forward for the next several lessons.