Pattern Matching with Match

In this lesson, you will learn about pattern matching and how it is implemented using the `match` control structure.

We'll cover the following

Pattern Matching

Pattern matching means exactly what you think it might—checking if an object or series of tokens match a specified pattern. A **pattern match** consists of a list of alternative cases which are made up of a pattern and a corresponding expression.

Using `match`

In Scala, *pattern matching* is done using the `match` expression. It is evaluated by taking the object to be matched and comparing it with each pattern in the order they are listed. The first pattern to match the object expression is selected and the corresponding expression is evaluated.

Syntax

```
selector match {
  case pattern => expression
  case pattern => expression
...
}
```

`selector` is our object to be matched with the list of alternative patterns. Each

`case` consists of a pattern followed by `=>` which is further followed by an expression to be evaluated if its corresponding pattern is selected.

An Example

Let's look at an example where given a food item, `match` matches it to its companion food.

> The code below, requires you to insert an input, the method of which has been discussed in a previous lesson.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val food = scala.io.StdIn.readLine()

food match {
  case "fish" => println("chips")
  case "peanut butter" => println("jelly")
  case "hamburger" => println("french fries")
  case _ => println("Try a new food!")
}
```

Pretty cool! When we give the match expression a food item, it tells us what we should pair with it. `_` is known as a **wildcard** pattern and matches with any object.

The purpose of this lesson was to provide you the syntax for using `match`. You will see the true power of pattern matching when we go over the different types of patterns provided by Scala.

In the next lesson, we have a challenge for you to solve for yourself.