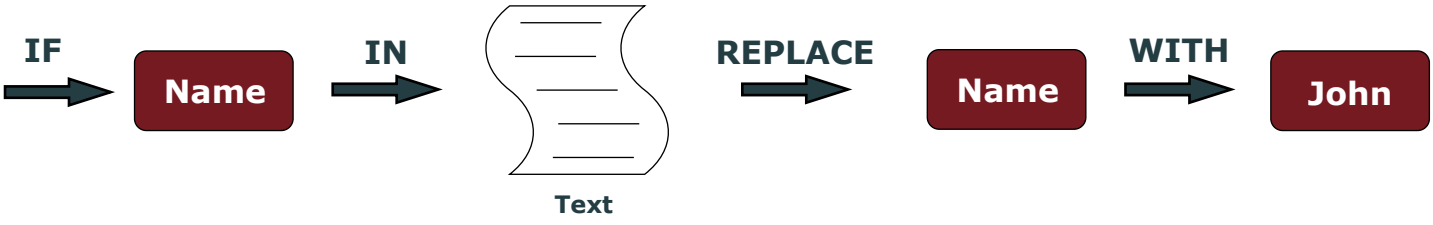
Replacing Patterns in Strings

In the following lesson, you will learn how to replace patterns in a string.

We'll cover the following

Problem

Imagine a situation where you not only want to find an expression, as we did in the previous lesson, but also want to replace it with another expression. For instance, let's say you're filling out an online form and where it says **name**, you'll need to replace it with your actual name.



Solution

One thing you need to know before we get into the solution is that strings are immutable, i.e., they cannot be modified. To overcome this obstacle, we create a new string which contains the replaced expression rather than modifying the old one.

> The built-in Scala methods we will discuss below, automatically create a new string, so you don't have to.

Replace the First

First, let's cover the solution for only replacing the first matching occurrence of an expression. Scala has two inbuilt methods that we can use. Both produce the same output but follow a different syntax.

`replaceFirst`

`replaceFirst` is called on the string that needs to be modified either directly or using a string variable.

```
String Literal.replaceFirst("SearchExpression","ReplaceExpression")
```

OR

```
VariableName of String.replaceFirst("SearchExpression","ReplaceExpression")
```

`SearchExpression` is where you would insert the expression you want to replace and `ReplaceExpression` is where you would insert the expression you want to replace the previous expression with.

Let's look at an example where we need to replace the first **0** or **1** in the string with a capital **X**.

The regular expression which represents the patterns *0* and *1* is `"[01]"`. This is representing a set of characters namely **0** and **1**.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val replaceIn = "8201530"
val replaced = replaceIn.replaceFirst("[01]", "X")

// Driver Code
println(replaced)
```

`replaceFirstIn`

Unlike `replaceFirst`, `replaceFirstIn` is called on the regular expression that needs

to be replaced rather than the string that needs to be modified.

```
Regex.replaceFirstIn("SearchString","ReplaceExpression")
```

As before, `ReplaceExpression` is where you would insert the expression you want to replace the previous expression with. `SearchString` is where you would insert the string to be modified.

Let's look at an example where we need to replace the first **H** in a string with **J**.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val regularExp = "H".r

val replaceIn = "Hello World!"

val replaced = regularExp.replaceFirstIn(replaceIn,"J")

// Driver Code
println(replaced)
```

Replace them All

Now, we will move on to methods which replace all matching occurrences of a regular expression with a new expression. Again, Scala has two inbuilt functions which do exactly that and we will discuss each one below.

`replaceAll`

`replaceAll` is called on the string that needs to be modified either directly or using a string variable.

```
String Literal.replaceAll("SearchExpression","ReplaceExpression")
```

OR

```
VariableName of String.replaceAll("SearchExpression","ReplaceExpression")
```

The syntax is identical to `replaceFirst` where `SearchExpression` is the expression you want to replace and `ReplaceExpression` is the expression you want to replace the previous expression with.

Let's build upon the example for `replaceFirst`, but this time we want to replace all the **0**'s and **1**'s with **X**'s.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val replaceIn = "8201530"
val replaced = replaceIn.replaceAll("[01]", "X")

// Driver Code
println(replaced)
```

`replaceAllIn`

Just as `replaceFirstIn`, `replaceAllIn` is called on the regular expression that needs to be replaced rather than the string that needs to be modified.

```
Regex.replaceAllIn("SearchString","ReplaceExpression")
```

The syntax is identical to `replaceFirstIn` where `ReplaceExpression` is the expression you want to replace the previous expression with and `SearchString` is the string to be modified.

Let's look at an example where we have a string of letters and numbers, but only want a string of numbers. What we will do is replace all the blocks of letters with the number **1**.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val regularExp = "[a-z]+".r

val replaceIn = "dk79rx5c4lj2c8ge"

val replaced = regularExp.replaceAllIn(replaceIn,"1")

// Driver Code
println(replaced)
```

With this lesson, our `String` manipulation problems come to an end. Let's finish our discussion on strings with `String` comparisons in the next lesson.