String Interpolation with 's'

In the following lesson, you will be introduced to the 's' string interpolator.

We'll cover the following

Processed String Literals

In string interpolation, we work with *processed* string literals. Processed string literals are simply string literals that require further work or *processing* during compilation by the compiler.

The `s` String Interpolator

For string interpolation with `s`, we prepend an **s** to any string literal. This allows us to use variables inside a string.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val country = "Japan"

println(s"I want to visit $country!")
```

In our example above, `s"I want to visit $country!"` is a processed string literal.

Syntax for Single Variable Expressions

```
s"Optional String $VariableIdentifier Optional String"
```

You can place any expression after the `$`. For a single variable expression, you can simply place the variable name after the `$` and Scala will interpret all the characters until the last character of the variable identifier.

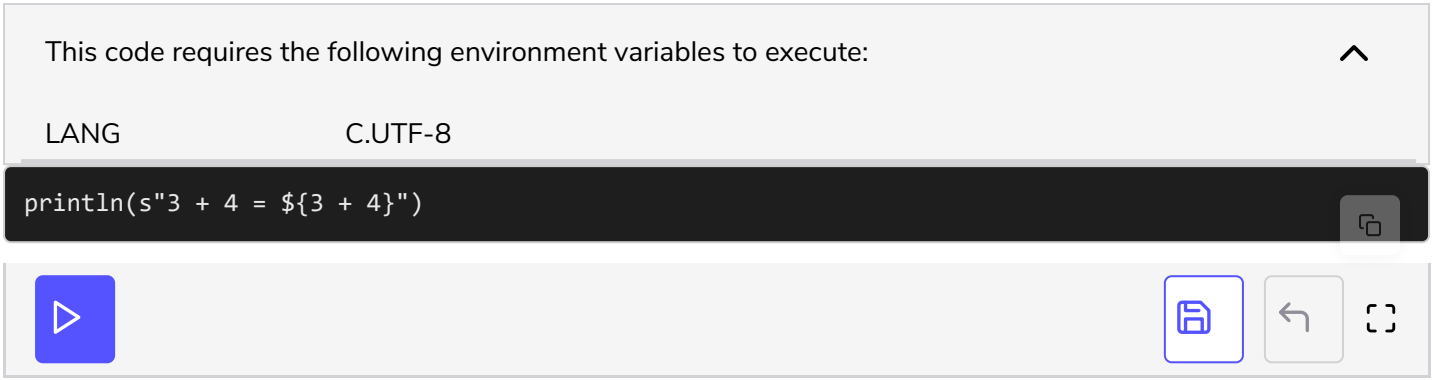
In our example above, the variable identifier was `country`. Scala interpreted the variable until the `y` of `country` and afterward, processed `!` like any other character.

Let's now try to embed a mathematical expression in a string. As a mathematical expression doesn't have any identifiers, letting the compiler know what to interpret and what to take as a regular character is done by modifying the syntax for single variable expressions.

Syntax for Expressions with Non-Identifier Characters

```
s"Optional String ${Expression} Optional String"
```

The expression you want the Scala compiler to process will be placed in curly brackets. The opening curly bracket will immediately follow the `$`.



In the code above, `3+4` is our expression which the compiler processes and interprets to `7`, as can be seen in the output when you press **RUN**.

Feel free to modify the above code snippets to get a better hang of string interpolation with `s`.

In the next lesson, you will be challenged to embed a variable in a string.