

Strings and Other Literals

In the following lesson, you will be introduced to literals.

We'll cover the following

- Literals
 - String Literals
 - Integer Literals
 - Floating-Point Literals
 - Character Literals
 - Boolean Literals

By now, you must have noticed that we haven't discussed strings. Strings do not fall under `anyVal` in the data type hierarchy. This raises the question: what are strings?


The single word answer is **'literals'**.

Literals

A literal is defined as taking anything in its most usual and basic sense. Mapping this onto computer programming, literals are fixed values appearing directly as is in the source code. For example, "Hello World", 5, and 'A' are all literals.

String Literals

Strings literals are a combination of characters surrounded by double quotation marks. The syntax for declaring a variable of type `String` is the same as declaring a variable of any basic value type.

This code requires the following environment variables to execute: 

LANG C.UTF-8

```
val stringLiteral: String = "Hello"
```

```
// Driver Code
```



```
println(stringLiteral)
```



Integer Literals

Integer Literals are used with the value types `Long`, `Int`, `Short`, and `Byte` and come in two forms: decimal and hexadecimal. Decimal numbers have a base 10 while hexadecimal numbers have a base 16. The way an integer literal begins indicates the base of the number. If the number begins with a `0x` or `0X`, it is hexadecimal and may contain the numbers from 0 through 9 as well as upper or lowercase letters from `A` to `F`. Numbers that do not start with `0x` or `0X` are decimal by default.

This code requires the following environment variables to execute:



LANG

C.UTF-8

```
val hex: Int = 0x0F5
val dec: Int = 245

val hex1: Long = 0x0A3DE5L //The L at the end is for Long
val dec1: Long = 671205L

// Driver Code
println(hex)
println(dec)
println(hex1)
println(dec1)
```



One thing to remember is that Scala will always print an integer literal as decimal regardless of how it is declared.

Floating-Point Literals

Floating-point literals are used with `Double` and `Float`. They are made up of decimal digits and have the option of adding exponents using either an `e` or `E` followed by the exponent.

This code requires the following environment variables to execute:



LANG

C.UTF-8

```
1. float: Float = 1.2345678901234567E-05 // The E at the end is for Float
```

```
val floatLiteral: Float = 1.2345F // The F at the end is for Float
val somethingBigger: Double = 1.2345e1
val evenBigger: Double = 1.2345e4
```

```
// Driver Code
println(floatLiteral)
println(somethingBigger)
println(evenBigger)
```



Character Literals

Character literals are used with `Char` and are made up of a single Unicode character surrounded by single quotation marks.

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val charLiteral: Char = 'A'
val smile: Char = '😊'
```

```
// Driver Code
println(charLiteral)
println(smile)
```



Escape sequences, as shown in the table below, also represent character literals.

Literal	Meaning
<code>\n</code>	linefeed
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\f</code>	form feed
<code>\r</code>	carriage return





"	double quote
'	single quote
\	backslash

Let's look at some examples:

This code requires the following environment variables to execute: ^

LANG C.UTF-8

```
println("separate\nlines")    //using \n to end the line
println("tab\tbetween\twords") //using \t to insert a tab
println('\\')                 //using \\ to print a backslash
println('\''')                 //using \' to print a single quotation mark
println('\\"')                 //using \" to print a double quotation mark
```

Boolean Literals

Boolean literals are used with `Boolean`. They are of two types: `true` and `false`.

This code requires the following environment variables to execute: ^

LANG C.UTF-8

```
val theTruth: Boolean = true
val aLie: Boolean = false

// Driver Code
println(theTruth)
println(aLie)
```






Let's complete our discussion on variables and types with type inference in the next lesson.