Operator Precedence

In the following lesson, you will be introduced to which operators in Scala hold precedence over others.

We'll cover the following

- Introduction
- Precedence Table
  - Assignment Operators
- A Good Practice

Introduction

Operator precedence determines the order with which different parts of the code/expression should be evaluated. For instance, `1 + 1 * 5` would give us `7` rather than `10` as `*` has higher precedence than `+`. If we wanted `10` we could write the expression as `(1 + 1) * 5` as `()` has higher precedence than `*`.

By now we have concluded that operators in Scala aren't really operators, they are simply methods in operator notation. Hence, when looking at operator precedence, we look at the first character of the symbol of an operator (method). If the first character of an operator is higher up in the precedence table, it will be evaluated first.

Precedence Table

Below, you'll find the precedence with operator precedence being highest at the top and getting lower as you come down.

Operator Precedence
`()` `[]`
special characters: `~` `!`

`*` `/` `%`

`+` `-`

`:`

`!=` `==`

`<` `>`

`&`

`^`

`|`

letters

assignment operators

There might be some operators in the list you aren't familiar with, but that's okay; they will be discussed in future chapters. The purpose of this list is to provide you a comprehensive operator precedence order which you can go back to again and again whenever needed.

Assignment Operators

Assignment operators do not follow the conventional rule of the first character preference. Rather a method which is followed by an *equal* sign will have the same precedence as the assignment operator `=`. The exception to this rule is comparison operators such as `<=` and `!=` which will follow the first character rule as discussed above.

A Good Practice

While operator precedence is a valid and supported part of Scala, it is still better to show precedence using parenthesis `()`. This avoids any and all confusion that a programmer reading your code might have.

And on this note, our discussion on operators comes to an end. In the next lesson, you will be challenged to use your knowledge on operator precedence.