INFO0054 Programmation Fonctionnelle – Exercises

Christophe Debruyne

Exercises 01: Introduction to Scala

We assume that you have installed Scala and an IDE of your choice. Throughout these exercises, we will use MS Visual Studio. We also assume that you are familiar with a terminal.

Exercise 1:

In this exercise, we will run our first Scala program by copy and pasting some code. Create the file `TP1.scala`, and copy and paste the following code inside that file. You may omit the curly braces as long as you respect the indentation rules (much like Python). If you omit the curly braces, ensure that the last line is a new line without any white space characters (i.e., a carriage return + new line).

```
def square(x: Int): Int = {
    x * x
}

def cute(x: Int): Int = {
    x * x * x
}
```

1. Open the Scala REPL (Read-Eval-Print Loop) environment from the command line and load this file by executing: `:load TP1.scala`
2. Try invoking the functions `square` and `cube`.
3. Now reset the REPL environment by invoking `:reset`. Can you still invoke the function of your file? Why is that?
4. Quit the REPL environment by invoking `:quit`.

Exercise 2:

In this exercise, you will modify `TP1.scala`. Place the functions cube and square in a module `TP1`. First, load the file and invoke the functions from that module. Now try importing these function definitions from that module into the REPL environment so that you do not need to prefix them with their module. Do not forget to reset your REPL environment if you start from the previous exercise!

Exercise 3:

Create the function `roots` that takes as input the coefficients a, b, and c of a quadratic polynomial $ax^2 + bx + c$. `roots` returns a list of roots given by the formula $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. If $b^2 - 4ac < 0$, then there are no roots. If $b^2 - 4ac = 0$, then there is only one root. In all other cases, there are two roots.

Exercise 4:

(Challenge) Define `sumInt1`, a recursive function that takes as argument a natural number (positive integer in Scala) n and returns the sum of all natural numbers lower than or equal to n. Is your definition tail recursive? Justify your answer. If `sumInt1` is tail-recursive, define a version `sumInt2` that is recursive, but not tail recursive. If `sumInt1` is recursive, `sumInt12` should use a local tail-recursive function.

References

[1] Paul Chiusano and Rnar Bjarnason. 2015. Functional Programming in Scala (2nd. ed.). Manning Publications Co., USA.