

# ASSIGNMENT 1 – INTRODUCTION

## *Advanced programming paradigms*

In this serie, you will start by exploring the various tools at your disposal for developing functional programs. You will also be able to check that your tools have been installed correctly. In the second part of the serie, you will apply some of the knowledge you got during the first lesson in a practical exercise.

### Question 1 – Using *ScalaIDE*

- (a) Launch the ScalaIDE and create a new Scala project using *File* → *New* → *Scala Project*.
- (b) Within ScalaIDE, you can develop “normal” Scala code with classes etc. . . but this will come later on.

At the moment, we will focus on the usage of *worksheets*. Worksheets are similar to the REPL, except that you have access to multiple commands at the same time. Evaluation takes place every time you save the file and the result of the evaluation is displayed on the right-hand side of the screen (as depicted below):

```
1  def foo = 5.0           //> foo: => Double
2  def bar = 3             //> bar: => Int
3  def sum(x: Double, y: Double) = x + y    //> sum: (x: Double, y: Double)Double
```

Note that the evaluation results are actually comments in Scala (the rule for comments are similar to C++ or Java). If required, you can force the evaluation of the worksheet by pressing the keys **Ctrl** + **Shift** + **B** together.

- (c) Add a new worksheet (shortcut is **Alt** + **Shift** + **N**) called *FirstSteps*. Note that you can safely ignore the object declaration at the top of the screen now, it will be explained later.
- (d) Run the examples from slide 72 and 73 in the REPL. Try to make yourself at ease by defining other things.
- (e) In the worksheet, define a function that returns the square of a value. Check that your function works correctly by applying it to various values.
- (f) Define another function that returns the 4th power of a value, using the square function you just defined.
- (g) As you can see, the worksheet always returns the type that has been inferred for the expression you type or from the evaluation. What do you expect the worksheet to return for the following definition?

```
def bar(x: Int, y: Boolean) = "Hello"
```

.....

.....

.....

△ Turn page →

## Question 2 – *Getting our hands dirty*

You are now asked to write a function to compute the square root of a number<sup>1</sup>. Its prototype should be

```
1 def sqrt(x: Double) : Double
```

### 1. The Newton's method

A typical numerical method to compute the zeroes (or roots) of a function is the Newton's method. Given a function  $f$  and its derivative  $f'$ , we begin with a guess  $x_0$  for the root. A better approximation  $x_1$  of the root is then given by :

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \quad (1)$$

The process is then repeated with the recursion equation:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (2)$$

and stopped when the residual  $\epsilon$  is small enough.

### 2. Application to the square root function

Let's say one wishes to compute<sup>2</sup> `sqrt(612)`. This is equivalent to  $x^2 = 612$ . The function to use in Newton's method is then  $f(x) = x^2 - 612$ . Its derivative is  $f'(x) = 2x$ . With an initial approximation of 10 (you can choose what you want here), the steps are then :

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} = 10 - \frac{10^2 - 612}{2 \cdot 10} = 35.6 \\ x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} = 35.6 - \frac{35.6^2 - 612}{2 \cdot 35.6} = 26.3955 \dots \\ &\vdots \\ x_5 &= 24.73863375 \dots \end{aligned}$$

### 3. Implementation

As you can see, with only five steps the solution is already accurate to more than five decimal places (all the decimals written are correct). With the help of recursion, you now have to implement this method for computing square roots.

- Create a new worksheet in Eclipse to write your code for this assignment.
- Define a function `isGoodEnough` that determines if your solution is good enough. Your solution can be considered good enough for example when  $\epsilon < 0.0001$ . For this part, you need to compute an absolute value function.
- Define another function, called `improve`, to compute the value of  $x_{n+1}$ , given the current approximated value and the value of  $x$ .
- Using the previously defined functions, define the `sqrt` method. Please note that you can add other functions if you need to!
- Test your method and check your results.
- [Optional] Implement the cubic root using the same approach and check your results.

<sup>1</sup>This exercise is originally from the SICP

<sup>2</sup>Example from [http://en.wikipedia.org/wiki/Newton's\\_method](http://en.wikipedia.org/wiki/Newton's_method)