Example: Square roots with Newton's method

Principles of Functional Programming

Task

We will define in this session a function

```
/** Calculates the square root of parameter x */
def sqrt(x: Double): Double = ...
```

The classical way to achieve this is by successive approximations using Newton's method.

Method

To compute `sqrt(x)`:

- Start with an initial *estimate* `y` (let's pick `y = 1`).
- Repeatedly improve the estimate by taking the mean of `y` and `x/y`.

Example:

Estimation	Quotient	Mean
1	2 / 1 = 2	1.5
1.5	2 / 1.5 = 1.333	1.4167
1.4167	2 / 1.4167 = 1.4118	1.4142
1.4142	...	...

Implementation in Scala (1)

First, define a function which computes one iteration step

```
def sqrtIter(guess: Double, x: Double): Double =
  if isGoodEnough(guess, x) then guess
  else sqrtIter(improve(guess, x), x)
```

Note that `sqrtIter` is *recursive*, its right-hand side calls itself.

Recursive functions need an explicit return type in Scala.

For non-recursive functions, the return type is optional

Implementation in Scala (2)

Second, define a function `improve` to improve an estimate and a test to check for termination:

```
def improve(guess: Double, x: Double) =
  (guess + x / guess) / 2

def isGoodEnough(guess: Double, x: Double) =
  abs(guess * guess - x) < 0.001
```

Implementation in Scala (3)

Third, define the `sqrt` function:

```scala
def sqrt(x: Double) = sqrtIter(1.0, x)
```

Exercise

1. The `isGoodEnough` test is not very precise for small numbers and can lead to non-termination for very large numbers. Explain why.
2. Design a different version of `isGoodEnough` that does not have these problems.
3. Test your version with some very very small and large numbers, e.g.

```
0.001
0.1e-20
1.0e20
1.0e50
```