Learning by Example: Newton's Method

In this lesson, you will learn how to calculate the square root of a number using Newton's Method. We will then go over its implementation in Scala.

We'll cover the following

In this lesson, we will define a function which will calculate the square root of a given number. The implementation of the function will be based on Newton's Method.

Newton's Method

Newton's method is used for approximating solutions. It is an iterative process and starts with an initial estimate which is used to find a better approximation which is then further used to find a better approximation until we reach the required result.

Computing the Square Root of x

To compute the square root of a number, **x**, we first need to make an initial estimate, **y**, let's say **1**. We improve the estimate by taking the average of **y** and **x/y**.

Let's look at an example. The table below shows how Newtons method would compute the square root of **4** by approximation.

Estimate(y)	Quotient(x/y)	Average
1	4/1=4	2.5
2.5	4/2.5=1.6	2.05
2.05	4/2.05=1.9512195122	2.0006097561
2.0006097561	4/2.0006097561=1.9993904297	2.0000000929

If we continue the above calculations, the average would eventually converge to **2**.

Brace yourself because below, you'll be introduced to your first extensive Scala program!

Implementation in Scala

To implement the above method in Scala, we need to define five functions interdependent with each other.

- **abs()**: will return the absolute value of a given number
- **isGoodEnough()**: will let us know if the average is close enough to the actual value
- **improve()**: will return the average of **y** and **x/y**
- **sqrtIter()**: a recursive function which will compute each iteration of Newton's method
- **sqrt()**: Will use the **sqrtIter()** function and return the square root of the given number

> When defining a function dependent on other functions, make sure to define them above the dependent function.

```
def abs(x: Double) =
  if (x < 0) -x else x

def isGoodEnough(guess: Double, x: Double) =
  abs(guess * guess - x) / x < 0.0001

def improve(guess: Double, x: Double) =
```

```
  (guess + x / guess) / 2

def sqrtIter(guess: Double, x: Double): Double =

  if (isGoodEnough(guess, x)) guess
  else sqrtIter(improve(guess, x), x)

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

To compute the square root of a number, all we have to do is call the `sqrt` function. `sqrt` is dependent on `sqrtIter` which is further dependent on both `improve` and `isGoodEnough`. `isGoodEnough` is then dependent on `abs`.

> Take some time here and go over the program above before moving on to the next lesson.

Let's call `sqrt` and see what happens.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
def abs(x: Double) =
  if (x < 0) -x else x

def isGoodEnough(guess: Double, x: Double) =
  abs(guess * guess - x) / x < 0.0001

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

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

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

val result = sqrt(4)

println(result)
```

Let's use the example you learned in this lesson to learn about **blocks** in the next lesson.