First steps in Scala

Like Python, Scala has an interactive mode, where you can try out things or just compute something interactively.

```
Welcome to Scala version 2.8.1.final.

scala> println("Hello World")
Hello World

scala> println("This is fun")
This is fun
```

To write a Scala script, create a file with name, say, test.scala, and run it by saying `scala test.scala`.

In the first homework you will see how to compile large programs (so that they start faster).

Typing

What is the biggest difference between Python and Scala?

Python is dynamically typed, Scala is statically typed.

In Python and in Scala, every piece of data is an object. Every object has a type. The type of an object determines what you can do with the object.

Dynamic typing means that a variable can contain objects of different types:

```
# This is Python
def test(a, b):
  print a + b

test(3, 15)
test("Hello", "World")
```

The + means different things.

Static typing

Static typing means that every variable has a known type.

If you use the wrong type of object in an expression, the compiler will immediately tell you (so you get a compilation error instead of a runtime error).

```
scala> var m : Int = 17
m: Int = 17

scala> m = 18
m: Int = 18

scala> m = 19.0
<console>:6: error: type mismatch;
```


Dynamic and static typing

Dynamic typing: Flexible, concise (no need to write down types all the time), useful for quickly writing short programs.

Static typing: Type errors found during compilation. More robust and trustworthy code. Compiler can generate more efficient code since the actual operation is known during compilation.

Java, C, C++, and Scala are all statically typed languages. But Scala is the only modern language among them: Scala uses type inference, and you don't have to write types all over your program. (C++11 has the `auto` type.)

```
scala> var t = 18.0
t: Double = 18.0
```

val and var variables

Scala has two kinds of variables:

`val` variables can never change their value. After the variable name has been defined, it always keeps its current value:

```
scala> val u = 17
u: Int = 17
scala> u = 18
<console>:6: error: reassignment to val
```

`var` variables are like variables in Python—their value can change as often as you want.

It is easier to reason about programs if they use `val` variables.

The extreme case—programs without `var` variables—leads to a style called functional programming.

Functions without result

A function or expression that returns nothing useful returns the special value `()` of type `Unit` (similar to `None` in Python).

```
def greet(name: String) : Unit = {
  println("Hello " + name)
}
```

Since this case is quite common, Scala provides special syntax:

```
def greet(name : String) {
  println("Hello " + name)
}
```

No = sign!

Functions

In mathematics:

$$f : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}, \quad f(a, b) = a + b$$

In Scala:

```
def f(a: Int, b: Int) : Int = a + b
```

Argument types

Result type

Function definition

The function definition is a block: A single expression or a sequence surrounded by curly braces.

No `return` statement needed. A function returns its last expression value.

In Scala, every statement is an expression and returns a value.