Scala-tulkin saa avattua komentoriviltä ajamalla käskyn scala. Scala ohjelman saa käännettyä komennolla scalac [tiedosto] ja ohjelman saa ajettua käskyllä scala [tiedosto]. Suosittelen kuitenkin lataamaan Eclipsen ja asentamaan siihen scala-tuen.

Tulkissa voi suorittaa scala koodia, esim.:

```
scala> val msg = "Hello, world!" //val on immutable muuttuja
msg: java.lang.String = Hello, world!

scala> println(muu)
666

scala> var muu = 666 //Luodaan muuttuja muu
muu: Int = 666
```

Jos rakenne jää kesken sitä voi jatkaa:

```
scala> def max(x: Int, y: Int): Int = {
     | if (x > y) x
     | else y
     | }
max: (x: Int, y: Int)Int
```

Scalassa voi käyttää funktioita parametrina:

```
  def koristele(x: Int, y: Int, arvo: (Int, Int) => Int) =

    println("**** " + arvo(x, y) + " ****")

arvo on kuvaus kokonaislukuparilta kokonaisluvulle
```

Ja näin kätevästi sitä sitten voi käyttää:

```
  koristele(5, 9, (x: Int, y: Int)  => x + y)  // **** 14 ****
  koristele(5, 9, (x: Int, y: Int)  => x * y)  // **** 45 ****

  def erotus(a: Int, b: Int) = a-b
  koristele(5, 9, erotus)                      // **** -4 ****
```

Taulukko-olioita voidaan luoda ja käyttää seuraavaan tapaan:

```
  val greetStrings = new Array[String](3)

  greetStrings(0) = "Hello"
  greetStrings(1) = ", "
  greetStrings(2) = "world!\n"

  for (i <- 0 to 2) //0 to 2 luo Range olion
    print(greetStrings(i))
```

Muuta:

-funktioita voi määritellä missä vain lohkossa ja niitä käsitellään kuin muuttujia

-funktion nimessä voi käyttää erikoismerkkejä, joten voi vaikka tehdä luokalle funktion +=, jota voi myös käyttää näin: muuttuja += jotain ( voi myös kirjoittaa muuttuja.+=(jotain) )

-pääkonstruktorin parametrit tulevat luokan nimen jälkeen