We use **sbt for building, testing, running and submitting assignments**. This tutorial explains all sbt commands that you will use during our class. The Tools Setup page explains how to install sbt.

The following page introduces:

1. Basic sbt tasks useful for any Scala developer. For more details about SBT or their commands, we highly recommend you to check the SBT Reference Manual or this SBT tutorial made by the Scala Community.
2. A way to submit your assignments to our Coursera graders with SBT.

The basics

Base or project's root directory

In sbt's terminology, the "base or project's root directory" is the directory containing the project. So if you go into any SBT project, you'll see a `build.sbt` declared in the top-level directory that is, the base directory.

Source code

Source code can be placed in the project's base directory as with hello/hw.scala. However, most people don't do this for real projects; too much clutter.

SBT uses the same directory structure as Maven for source files by default (all paths are relative to the base directory):

```
src/
  main/
    resources/
       <files to include in main jar here>
    scala/
       <main Scala sources>
    java/
       <main Java sources>
  test/
    resources
       <files to include in test jar here>
    scala/
       <test Scala sources>
    java/
       <test Java sources>
Other directories in src/ will be ignored. Additionally, all hidden
directories will be ignored.

```

Other directories in src/ will be ignored. Additionally, all hidden directories will be ignored.

SBT build definition files

You've already seen build.sbt in the project's base directory. Other sbt files appear in a project subdirectory.

The *project* folder can contain .scala files, which are combined with .sbt files to form the complete build definition. See organizing the build for more.

```
build.sbt
project/
  Build.scala

```

You may see .sbt files inside project/ but they are not equivalent to .sbt files in the project's base directory. Explaining this will come later, since you'll need some background information first.

SBT tasks

Starting up sbt

In order to start sbt, open a terminal ("Command Prompt" in Windows) and navigate to the directory of the assignment you are working on (where the *build.sbt* file is). Typing sbt will open the sbt command prompt.

```
# This is the shell of the operating system
$ cd /path/to/parprog-project-directory
$ sbt
# This is the sbt shell
>
```

SBT commands are executed inside the SBT shell. Don't try to execute them in the Scala REPL, because they won't work.

Running the Scala Interpreter inside SBT

The Scala interpreter is different than the SBT command line.

However, you can start the **Scala interpreter inside sbt** using the `*console*` task. The interpreter (also called REPL, for "read-eval-print loop") is useful for trying out snippets of Scala code. When the REPL is executed from SBT, all your code in the SBT project will also be loaded and you will be able to access it from the interpreter. That's why the Scala REPL can only start up if there are no compilation errors in your code.

In order to quit the interpreter and get back to sbt, type *<Ctrl+D>*.

```
> console
[info] Starting scala interpreter...
Welcome to Scala version 2.11.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7
  .0_04-ea).
Type in expressions to have them evaluated.
Type :help for more information.

scala> println("Oh, hai!")                                    # This is
  the Scala REPL, type some Scala code
Oh, hai!

scala> val l = List(1, 2, 3)
l: List[Int] = List(1, 2, 3)

scala> val squares = l.map(x => x * x)
squares: List[Int] = List(1, 4, 9)

scala>                                                        # Type [ctrl
  -d] to exit the Scala REPL
[success] Total time: 20 s, completed Mar 21, 2013 11:02:31 AM
>                                                             # We're back
  to the sbt shell
```

Compiling your Code

The compile task will compile the source code of the assignment which is located in the directory src/main/scala.

```
> compile
[info] Compiling 4 Scala sources to /Users/aleksandar/example/target/scala-2.11
  /classes...
[success] Total time: 1 s, completed Mar 21, 2013 11:04:46 PM
>
```

If the source code contains errors, the error messages from the compiler will be displayed.

Testing your Code

The directory src/test/scala contains unit tests for the project. In order to run these tests in sbt, you can use the test command.

```
> test
[info] ListsSuite:
[info] - one plus one is two
[info] - sum of a few numbers *** FAILED ***
[info]   3 did not equal 2 (ListsSuite.scala:23)
[info] - max of a few numbers
[error] Failed: : Total 3, Failed 1, Errors 0, Passed 2, Skipped 0
[error] Failed tests:
[error]   example.ListsSuite
[error] {file:/Users/luc/example/}assignment/test:test: Tests unsuccessful
[error] Total time: 5 s, completed Aug 10, 2012 10:19:53 PM
>
```

Running your Code

If your project has an object with a main method (or an object extending the trait App), then you can run the code in sbt easily by typing run. In case sbt finds multiple main methods, it will ask you which one you'd like to execute.

```
> run
Multiple main classes detected, select one to run:

 [1] example.Lists
 [2] example.M2

Enter number: 1

[info] Running example.Lists
main method!
[success] Total time: 33 s, completed Aug 10, 2012 10:25:06 PM
>
```

Submitting your Solution to Coursera

The sbt task submit allows you to submit your solution for the assignment. It will pack your source code into a .jar file and upload it to the coursera servers. Note that the code can only be submitted if there are no compilation errors.

Warnings: Before proceeding:

- Make sure that you run sbt in the root folder of the project (where the *build.sbt* file is).
- Make sure that the console line is `>` and not `*scala*>`. Otherwise, you're inside the Scala console and not the sbt shell.

The submit tasks takes two arguments: your Coursera e-mail address and the submission token. **NOTE**: the submission token is **not your login password**. Instead, it's a special password generated by coursera for every

assignment. It is available on the Assignments page.

```
> submit e-mail@university.org suBmISsioNPasSwoRd
[info] Packaging /Users/luc/example/target/scala-2.11/parprog-example_2.11-1.0.0
  -sources.jar ...
[info] Done packaging.
[info] Compiling 1 Scala source to /Users/luc/example/target/scala-2.11/classes
  ...
[info] Connecting to coursera. Obtaining challenge...
[info] Computing challenge response...
[info] Submitting solution...
[success] Your code was successfully submitted: Your submission has been
  accepted and will be graded shortly.
[success] Total time: 6 s, completed Aug 10, 2012 10:35:53 PM
>
```

You can also submit your work without starting the sbt interactive shell, by running the following command:

```
$ sbt "submit e-mail@university.org suBmISsioNPasSwoRd"
```

Note the usage of double quotes.

Mark as completed