注意本課講課的人就是網上說的那個写 权威艰涩 Scala 教程 的作者 Martin Odersky

IntelliJ IDEA

用 IntelliJ IDEA 寫 scala 程序之步驟:
啟動 IntelliJ IDEA:
cd /home/tao/IntelliJ-IDEA/idea-IC-171.4249.39/bin
./idea.sh
(No longer works: 在 Ubuntu 中搜 IntelliJ 即可啟動, 即在搜 pdfsam 那裡.)
Create New Project
Choose Scala, choose SBT. Next.
填 Project Name(比如 Example). JDK version 選 1.8. Finish.
等一會兒出現新窗口, 在新窗口中點左上角的 Project 的名字(比如 Example)左邊的三角形, 選文件夾 src -> main

Optional: 寫一些簡單的 scala 語句並運行(這樣生成的文件為 HelloWorld.sc):
右鍵點 main 中的 scala → New → Scala Worksheet → File name(如 HelloWorld).
在新窗口中輸入程序語句, 如就一行(寫的時時要加引號): "Hello World!"
然後會自動編譯(不用我做甚麼).
然後按上面的那個 綠色向右的三角箭頭, 即可運行. 結果會顯示在右邊的窗口.

寫正常的程序(這樣生成的文件為 Example.scala):
右鍵點 main 中的 scala → New → Scala Class → 填 Name(比如 example.Example, 後面的 Example 即 Project 名字), Kind 選 Object -> OK.
然後寫代碼, 要 extend App, 例如以下為一個可以運行的完整代碼:

```
package example

object Exmple extends App {
   println("Hello World")  //沒有分號
}
```

然後點 object Exmple extends App 左邊的那個按鈕(像兩個長方形), 有三個選項:
Run 'Example'
Debug 'Example'
Run 'Example' with Converage
選 Run 'Example', 即可運行程序. 然後在下面可以看到結果.

打開一個 SBT project:
左上角: File -> Close Project. 代碼窗口就被關了.
在小點的窗口中點 Import Project.
然後出來一個小窗口, 選路徑/home/tao/IdeaProjects -> Example -> target -> build.sbt -> OK, 出現一個新窗口, 可以選 SDK 的版本等. 點 OK. 問是否 overwrite, 選 Yes.
然後出來一個小窗口, 窗口名字叫 SBT Project Data To Import, 直接即 OK.
在打開的代碼窗口中, 耐心等, 一會兒後, 會在左邊出現 Project 文件結構, 點 Example -> target -> 雙擊 build.sbt (雙擊兩個字打出來不容易, 電腦反覆死機了好幾次). 出來的 build.sbt 代碼窗口中有以下三行: name, version, scalaVersion.
如果代碼右上角出現三個選項: Refresh project, Enable auto-import, Ignore, 則點 Enable auto-import.

不要在下面加 tutorial 中的一句:
libraryDependencies += "org.scalatest" %% "scalatest" % "2.2.6" % "test"

真正的代碼就在 src/main/scala 中.

然後就可以運行之前寫的代碼了,
在 src->main->scala->example 中可以找到之前寫的 HelloWord.sc, 即只有一句"Hello World!"那個, 然後, 可用前面一樣的方法運行. 在同一個文件夾中也可以找到之前寫的 object(即寫了 extend App 的那個), 也可以按前面一樣的方法運行.

以下 terminal 沒甚麼卵用:
The easiest way to run SBT commands from from IntelliJ IDEA is to use the Terminal tool window via Alt+F12. It executes the terminal in your project directory.
然後在 terminal 中用以下命令:
sbt
clean
compile

Running your Code

Once you start writing some code, you might want to run your code on a few examples to see if it works correctly. We present two possibilities to run the methods you implemented.

Using the Scala REPL (沒試的)

In the sbt console (sbt 和 REPL 用法見後面), start the Scala REPL by typing console.

```
> console
[info] Starting scala interpreter...
scala>
```

The classes of the assignment are available inside the REPL, so you can for instance import all the methods from object Lists:

```
scala> import example.Lists._
import example.Lists._
scala> max(List(1,3,2))
res1: Int = 3
```

Using a Main Object

Another way to run your code is to create a new Main object that can be executed by the Java Virtual Machine.

In IntelliJ:
right-click on the package example in src/main/scala and
→ New → Scala Class → Name 填 Main, Kind 選 Object -> OK.

In eclipse:
right-click on the package example in src/main/scala and

select “New” - “Scala Object”
Use Main as the object name (any other name would also work)
Confirm by clicking “Finish”

In order to make the object executable it has to extend the type App. Change the object definition to the following:

```
import example.Lists

object Main extends App {
  println(Lists.max(List(1,3,2)))
}
```

Now the Main object can be executed.

Execute in ItelliJ(就是將前面的步驟抄過來的, 實踐證明可以):
點 object Exmple extends App 左邊的那個按鈕(像兩個長方形), 有三個選項:
Run 'Example'
Debug 'Example'
Run 'Example' with Converage
選 Run 'Example', 即可運行程序. 然後在下面可以看到結果.

Execute in eclipse:
Right-click on the file Main.scala
Select “Run As” - “Scala Application”
You can also run the Main object in the sbt console by simply using the command run.

Writing Tests

Throughout the assignments of this course we will require you to write unit tests for the code that you write. Unit tests are the preferred way to test your code because unlike REPL commands, unit tests are saved and can be re-executed as often as required. This is a great way to make sure that nothing breaks when you have go back later to change some code that you wrote earlier on.

We will be using the ScalaTest testing framework to write our unit tests. In eclipse(或 IntelliJ), navigate to the folder src/test/scala and open the file ListsSuite.scala in package example. This file (已確認即 ListsSuite.scala 這個文件, “This file”這兩個單詞並非鍵接) contains a step-by-step tutorial to learn how to write and execute ScalaTest unit tests.

注意 ListsSuite.scala 這個文件中有這麼一句話:
Adding the `@RunWith` annotation enables the test suite to be executed inside eclipse using the built-in JUnit test runner.
所以 test 文件可能只能在 eclipse 中運行(沒確認).

SBT

During this class we will always launch the Scala REPL (the interactive Scala console) through sbt. This way you don't need to install the Scala distribution on your machine, having sbt is enough. (In

case you prefer to have the scala command available on your machine, you can download the Scala distribution from the scala-lang.org website.)

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:

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.
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-leve 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.

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>.

以下是 start scala interpreter:

```
> 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:

##
1
$ sbt "submit e-mail@university.org suBmISsioNPasSwoRd"
Note the usage of double quotes.

Getting Started

1. Course Introduction

Welcome. I'm going to give you now a quick introduction of my course, Functional Programming Principles in Scala, and tell you how we are going to organize it. As usual, the course will consist of a sequence of videos which will introduce the elements of the course one by one, and the videos will also have quizzes where you are asked to fill in some questions. Those questions could be multiple choice

or also, quite often, we ask you to program something. After the quizzes, you will always see the answer in the videos. Besides the quizzes, there are also assignments. The assignments you have to hand in, and you are going to be graded on the assignments. And if you pass the assignments, then you will get a certificate at the end of the course. The quizzes, by contrast, they are just for your own education. You are not going to be graded on them. There's a timeline for the course. You can expect a new set of videos every Tuesday morning, and including a new assignment. And you will have one week to hand in the assignments, or the assignments will have to be handed in Monday night the following week. We're going to use for the programming in the course the Eclipse (已裝), IDE, Integrated Development Environment. And, I will show you later on how you can download the IDE and how you can install it. There's also another tool we meet for handing in the assignments that's called SBT (已裝), the Scala Build Tool. Again, we will go through the details how to get and to install SBT. Okay. Once you are on the site, the first thing you could do is go to Tools Setup. Here, you'll see detailed setup instructions how to setup Eclipse and SBT for all the major operating systems, Windows, Mac OS and Linux. You could also simplify your life simply by looking at one of the three videos that can tell you exactly what to do to get started. Once you've done that, you could go next to Assignments, getting, going to get started in the assignments. And then you'll see an example assignment which is optional. We will grade it but it won't count towards your points for the course. That's essentially takes you through the steps of drafting an assignment, drafting an, an exercise submitting your solution to the assignment so that then when we start at the real assignments, you already know all the mechanics for that. That's all there is to know for now. Have fun with the course and the assignments.

2. Tools Setup for Linux (MacOS 和 Windows 的沒看)

(我檢查了的，前面沒有)System, so the cutting of the section in the video lectures part of the class website. Everything that is shown on this video, is also explained in text and tools of that page to the class website. So, we are here, and we need to install a few things for our class. The first one is the JDK, the Java development kit, just a random environment in which Scala programs are executed. This time the same as the [INAUDIBLE] which is strongly recommended to use this version over Java 7 and 6. The second one, this is SBT. The main build tool for Scala which we'll use for building your programs and submitting your assignments through Coursera. Last but not least, we need an IDE that will help us to go within Scala. In this case, we're going to show you how to install. There are other options, though, you can also use the Scala IDE and sign. Well, let's get started by installing JDK (已裝). Here there are three subsections for each operating system. As we are in Ubuntu, let's go to the right section. You have another Linux operating system. You can check which comments are necessary to install Java either in this website or in the official website offering the solution. In this tutorial ,we're only going to cover Ubuntu deviant systems. But, installing another system is pretty straightforward. Usually, this is done with pseudo app get, install, open JDK, 8JDK. As we have already installed Java, we are going to skip this process. But we're going to show you how this is done. So we go to the terminal, and we type pseudoapp.getinstall, open JDK8 JDK. Type our password and we'll see that in that place. And we have already installed this version. In case you need to check if you have the right Java version, you have to do java-version. And you type in the terminal and you press OK, enter. And you will see that the first line, that version that you have is in here. In this case, we have Java 8. Let's install SBT (已裝), in order to install SBT, first, we need to unload it. Installing SBT and we click this (http://www.scala-sbt.org/release/docs/Setup.html
或 https://www.coursera.org/learn/progfun1/supplement/BNOBK/tools-setup), this link which will refer you to this website. Actually, let's wait. Cool. So we go to this link. And now we have the instructions to install SBT in any system. So we go to the Linux link. And we'll see here, the commands that we need to execute, so that SBT is installed in our system. The first ones, are creating a new repository, while, having a new repository to the system. And, update it, so that we can install SBT with this simple comment. So, let's copy paste this comments. We go to the Terminal and we paste them. Once

SBT is installed, let's check that we have the right version of SBT. For that, we need to type “sbt about”. This is usually going to take some time on the first time you execute it because SBT has to bootstrap itself. As you see here, we do have the right version, 0.13.11. As a side note, SBT versions, are backwards compatible with previous versions of SBT, but not the other way around. That's good to know, when we use SBT in our daily workflow. So, now let's install IntelliJ IDEA (已装). For that, we need to go back to the tutorial, and we also go here, in this thing, to install this idea tutorial. So, First, we need to download Intellij IDEA. For that, we need to get to the official website and make sure that we are in the Linux option. And we download the IntelliJ IDEA. As we have already done this, we just need to go to the, Downloads folder. Oops, here, yeah. And we'll see that we have the compressed file that we have just downloaded and the file or the folder that is the result of decompressing this file. So usually you can just decompress this file with x + v + F6. And we're going to do it again. Cool. So we have the folder here. Right? And what we need to do is to go in, and we'll see that we have several folders. Bin, lib, plugins, etc. So what we need to do is to go to the binary folder, and we need to execute the script that's going to execute IntelliJ IDEA. That's called Idea.sh. The next step to install Esker plugin so the IntelliJ recognizes both SBT and Esker 5, so we go to configure And now we'll go to plugs ins here. And now we have to go here and install the plug in, and type. Usually, there is a green button here that says install. As we have already installed this car plugin, we don't need to do this. But you will, so the following steps for you will be to install escala and then restart [INAUDIBLE]. So we're going to close this, and now we're going to create a new project. In this window we have to make sure that we select this color and SBT. So, we're going to extend. Now we need to name our new project so let's name it "Sample". Let's make sure that we have the right versions. One eight here and zero thirteen and two eleven eight for SPTS color version. If you don't have anything here we need to go to type new and [INAUDIBLE] will automatically detect your [INAUDIBLE] the case, so just have to press okay and that's it. So we finish. With [INAUDIBLE]. And now [INAUDIBLE] is going to create the project. And it's going to select everything, so that we can start coding. Close this. Now SPT is executing. So, it's going to create a new source folder here and that's the one that we use for putting all our codes Once we are ready let's play around with this [INAUDIBLE] worksheet. This is a feature that gives you a very interactive console to work with this color. Now that we need to go to source and select main and go straight to discard folder. So, we just right click here and you here When need to give it name say, Hello World. And now we see that the format has changed, so the idea is that we can type anything we want to here, which is the [INAUDIBLE] command or [INAUDIBLE] I think only has Syntex. Valid Syntex, right, and it's going to compile it and it's going to show the result on the right. So, let's create here an as a strength and we'll see intelligent, well we press control S to save, then intellij is going to. It's going to automatically compile this and output results, which is going to be about Cool. So, now let's try to do something slightly more complicated, right? So, let's go here and let's create a main application in [INAUDIBLE] We need to write [INAUDIBLE] as well and we're going to create [INAUDIBLE] class. Our main application is going to be a [INAUDIBLE] here object and we're going to decide the name. This subject is going to be inside the package quote example, so we're going to type example and then dots. And then the name of our main application which is going to be example with [INAUDIBLE]. So, we press OK and here we see the result, but in order to allow [INAUDIBLE] compiler to run this main application, we need to extend. Application or apps APP, extends, APP. And now this is going to execute the body of this object, so it's pooling here in a statement that it's going to create hello world [SOUND]. As you see, [INAUDIBLE] has recognized this as the main application, and there is a small icon here that we can click. And we see that we have several options. The first one is run example, second one there is [INAUDIBLE] sample, and the third one is run example quick coverage. Here we're going to just run the example. And if everything goes well we're going to figure out the bottom part of the Call, so it has compiled. Now it's executing the program, and we see the output here. So let's go back to the tutorial. I'll see where we are. We're in the step seven, and now we're going to learn how to open an SBT project. We come back, we're going to

close the project and we're going to import it. By taking Q and input products. So, import a product A it goes here and we see that [INAUDIBLE] And in the project folder which is the project. So we select a built file and we click ok. And now, we see this window. And this window you can set up information about your project and specifically we see that we can set up the JAVA version, which is the correct one. One eight, and other options that we will explain later on. So we click there, and now SVT is going to create a new SVT project. But it's going to re-use the SVT project that we already created, but it has to protect the source files and also, which folders are in this project. We see the recognized folders. We continue. Now what we're going to do is change the beauty file. We find the beauty file here and we click on it. We can add a new dependency called scholar test as you see there is a notification here and this notification with three options. The first one is, reference project, the second one is enable IT port the other is to ignore. Enable and deport is official we will refresh SBT every time we change a file An SPT file or a SPT folder, this is extremely useful and I recommend to enable it. So we are going to enable it here, and we are going to add a dependency. So we are going to go back to the tutorial And copy, paste this command here, the second one. We'll save the file with control-S. And has automatically SVT. And now we are waiting for the result of SVT and the JSON error. Just a warning. Close and now [INAUDIBLE] is going to index the new appendices we just added to the project. Which is cool. It means that everything went fine. Now we want to execute any [INAUDIBLE] comment, we press alt 12 And these are going to open a terminal in our project folder with the example. Now we are able to execute this here. Remember that if you do this in another project folder, it's not going to work. It has to be in the work project folder. And we can type now nsbt Commands. So let's compare again the project. So that's it, now you're ready to use this current assignment. Good luck.