HACK >SOC

Learn you some

for greater good!

Luke Geeson

But what is scala?

Scala: The *scalable* language

Scala = Java + Functional Programming

Makes working concise, more so than java!

widely used in industry!

Why Scala?

- Runs on the Java JVM, so Java code and Scala can be run on the same stack. Compiles to Java Bytecode, so its pretty portable!
- Java and Scala are interoperable
- Static type system and automatic type inference
- in built asynchronous data handling and parallelisation (using Java-style futures and promises), oh and lazy evaluation! = scalability!
- Pattern matching: switch statements on steroids!
- Higher order functions and functional programming == expressibility

The history

- Started in 2001, by Martin Odersky, following work on funnel (another functional language)
- released publicly in 2004, on the java platform
- in 2011, Scala received €2.3 million from the European Research Council, allowing it to get commercial support
- used in industry a lot today!

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

A String

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

A function

A String

Hello, world!

A class/Object (this one is technically a ‘singleton’ object)

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

A function

A String

Files and some admin

Scala files are saved with `.scala` extensions,

e.g. `helloWorld.scala`

once Scala is installed, it comes with a ‘compiler’ to turn your `.scala` files into runnable byte code for the machine

this compiler is called `scalac`

compile your programs with: `scalac filename.scala`

run with `scala filename`

e.g. `scala HelloWorld`

Scala’s compilation model is identical to Java, and so you can use it with build systems like Ant or Gradle

Enter the Matrix

Comments

- Comments are the same in java:

```
// for single line comments

/*
   for block comments on multiple lines
*/
```

Variable Declaration

- Declare that variables exist with either `var` or `val`
- variables declared with `val` are **immutable**, that is they cannot change once set (like final in java)
- variables declared with `var` are **mutable,** for the java fans out there
- good practice to use `val` where you can

```
val num = 10
num = 20                        //error: reassignment to val
var meaningOfLife = 42
meaningOfLife = 34              //this is fine
meaningOfLife = 3.1415          //this is not, why?
```

Are you my type?

- Scala is **statically** typed, but it uses **type inference** too!

- static typing means it checks the variable types at **compilation** time so it doesn't have to when it runs (a bit like java, but not like python)

- Type inference means it automatically **guesses** the **types** of your expressions/statements and matches them accordingly, or **moans** at you if they are wrong.

  you can manually specify the type however with colons:

```
val numTriforceTriangles : Int = 3

val radius : Double = 33  //automatically converts types like these
```

some more types

- Spark has lots of inbuilt types, here are some more:
  Int
  Double
  Boolean
  Char

- Scala has a LOT of type stuff related to functional programming, if you're interested: https://twitter.github.io/scala_school/type-basics.html

Java is Spark is Java

- All of the standard class methods, and classes that come with **Java** can also be **directly used with Scala**, e.g the String class (and the syntax is the same):

```
"hello world".length()
"yolo swaggins".substring(4, 8)
```

- There is also some Scala specific methods, which are **functional** in nature

```
"hello world".take(4)  //takes the first 4 characters from the string
```

- See official Scala documentation for more

this is fun

- define functions like so:

```
def functionName(args...): ReturnType = { body... }

def fizzBuzz(x:Int){
   if (x % 3 == 0)
      println(“fizz”)
   else if (x % 5 == 0)
      println(“buzz”)
   else
      println(x)
}
```

- the **last** expression in the function block is the **return** value
- can **omit** the {} for the **function** block or **if** statements if they are **single** statements
- invoking functions is the **same** as in **every** language

more function stuff

- Give your arguments **default** values with '=':
  `def defaultsInMyFunc(x : Int = 4) = {...}`

- Make **anonymous** functions like so:
  `(number : Int) => number + 1`

- **or** like so:
  `val incr : Int => Int = _ + 1`

- check out https://www.coursera.org/course/progfun
  if you want to know about some crazy **functional** stuff

go with the flow

- If, while and do-while statements are the same as Java and C++

- you can specify a **range** of values using ‘to’

```
1 to 5
```

- you can **cycle** over these ranges using .foreach

```
(1 to 5).foreach(
   (number : Int) => println(number +1)
)
```

There’s more!

- Scala has **lots** of functional programming stuff, it has full support for **object oriented** programming too, I won’t go into detail as this could be a whole module in itself!
  checkout the resources!

- ON TO SPARK


HACK
>SOC

Useful resources(scala)

- Spark website: http://www.scala-lang.org/index.html
- Download Scala - http://www.scala-lang.org/downloads
- hello world: http://www.scala-lang.org/old/node/166
- learnxinyminutes: http://learnxinyminutes.com/
- some of my code here: https://gist.github.com/lukeg101/8af9e97fbb76bdf1dbdd