PEP Scala (1)

Email: christian.urban at kcl.ac.uk
Slides & Code: KEATS

Why Scala?

twitter

Linked in

the guardian

Morgan Stanley

CREDIT SUISSE

...

eDF ENERGY

Novell

foursquare

HSBC

...

A former student working now at Quantexa:

"I am a former student. I graduated last year. I got my dream job as a backend Scala developer. Most of the Scala I know is from PEP 2018/19. My interviewers said they expect code of a lesser quality even from people with one year of experience."

Why Scala?

- compiles to the JVM
  (also JavaScript, native X86 in the works)

- integrates seamlessly with Java

- combines **<u>functional</u>** and **object-oriented** programming

- no pointers, no null
- often one can write very concise and elegant code

Java vs Scala

```
public class Point {                              1
  private final int x, y;                         2
                                                  3
  public Point(int x, int y) {                    4
    this.x = x;                                   5
    this.y = y;                                   6
  }                                               7
                                                  8
  public int x() { return x; }                    9
                                                  10
  public int y() { return y; }                    11
}                                                 12
```
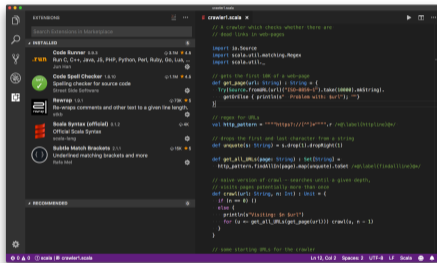
```
case class Point(val x: Int, val y: Int)
```

Scala

First Steps: Scala Tools

- contains a REPL

- I use VS Code and a Scala extension (M'place)

- there is a plugin for Eclipse (called Scala IDE)

- there is also a plugin for IntelliJ

My personal keboard shortcut for VS Code
(in keybindings.json)

```json
[
    {
        "key": "ctrl+enter",
        "command": "workbench.action.terminal.runSelectedText",
        "when": "editorTextFocus && editorHasSelection"
    }
]
```
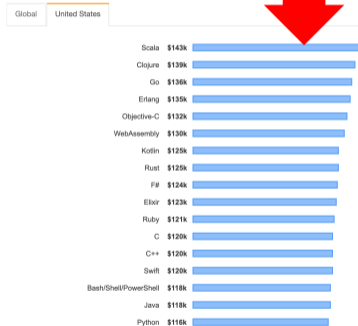
Why Scala?

Elm, Rust, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)…

Why Scala?

Money?

What Languages Are Associated with the Highest Salaries Worldwide?

Global | United States

Clojure	$90k
F#	$80k
Go	$80k
Scala	$78k
Elixir	$76k
Ruby	$75k
WebAssembly	$73k
Rust	$72k
Erlang	$71k
Bash/Shell/PowerShell	$69k
R	$64k
Python	$63k
Objective-C	$62k
TypeScript	$60k
C#	$59k
Swift	$59k
Kotlin	$57k

What Languages Are Associated with the Highest Salaries Worldwide?

Global | United States

Scala	$143k
Clojure	$139k
Go	$136k
Erlang	$135k
Objective-C	$132k
WebAssembly	$130k
Kotlin	$125k
Rust	$125k
F#	$124k
Elixir	$123k
Ruby	$121k
C	$120k
C++	$120k
Swift	$120k
Bash/Shell/PowerShell	$118k
Java	$118k
Python	$116k

* source: Stackoverflow Developer Survey, 2019

Elm, Rust, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)…

Why Scala?

Money?

Functional Programming!

* source: Stackoverflow Developer Survey, 2019

Elm, Rust, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)…

Why Functional Programming?

Elm, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)…

Why Functional Programming?

"If you want to see which features will be in mainstream programming languages tomorrow, then take a look at functional programming languages today."

—Simon Peyton Jones (works at Microsoft)
main developer of the Glasgow Haskell Compiler

Elm, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)...

Why Functional Programming?

Immutability

Elm, Haskell, Ocaml, F#, Erlang, ML, Lisp (Racket)…

Functions Procedures

INPUT INPUT

CHEEKY
ARBITRARY
EFFECTS

OUTPUT OUTPUT
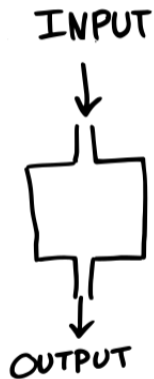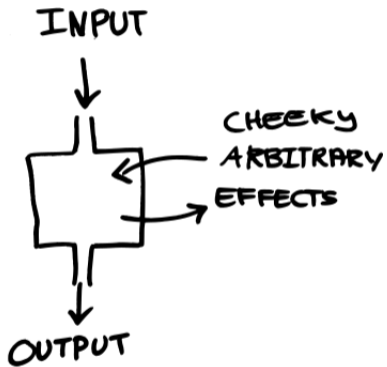
* from "What pure functional programming is all about?"

Why bother? or
What is wrong with this?

```
for (int i = 10; i < 20; i++) {

  //...Do something interesting
  //    with i...

}
```

1986

3 days

64K RAM, no HD, no monitor, lots of cables
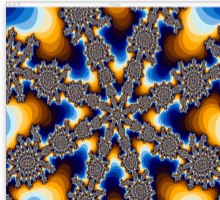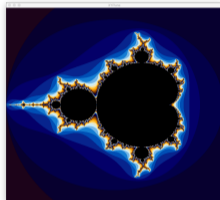
1986

1988, C

1986

1988, C

1992, Linux

1986

1988, C

1992, Linux

1996

1986

1988, C

1992, Linux

2000

1996

1986   1988, C   1992, Linux

2012?   2000   1996

1986

1988, C

1992, Linux

2012?

2000

1996

2017

1986

1988, C

1992, Linux

2012?

2000

1996

2017

1986: no Internet

no Amazon

no FB, no mobiles,...

1986
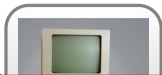
1988, C

1992, Linux

2012?

2017

Speedup by Moore's Law

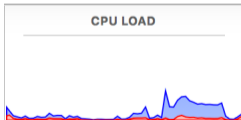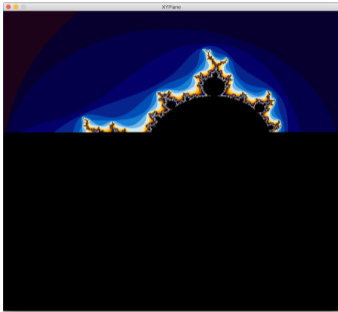|			
1986:	3 days	**1996:**	135 mins
1988:	1.5 days	**1998:**	67 mins
1990:	18 hs	**2000:**	33 mins
1992:	9 hs	**2002:**	16 mins
1994:	4.5 hs		???

Every two years, computers got twice as powerful.

no Amazon

no FB, no mobiles,...

1986

1988, C

1992, Linux

2012?

2000

1996

2017

1986: no Internet
no Amazon
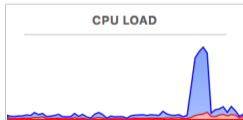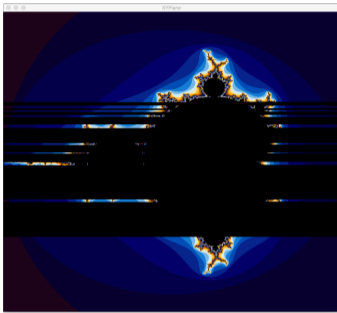**no FB, no mobiles,...**
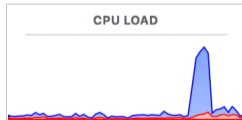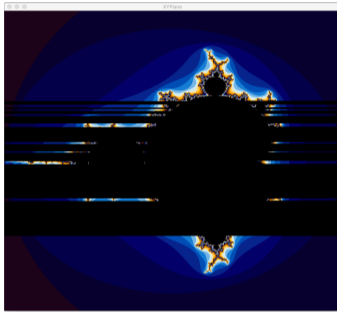
Seq vs Par

Seq vs Par
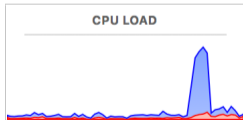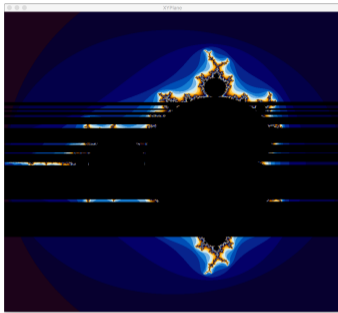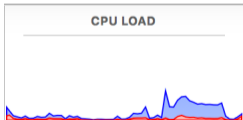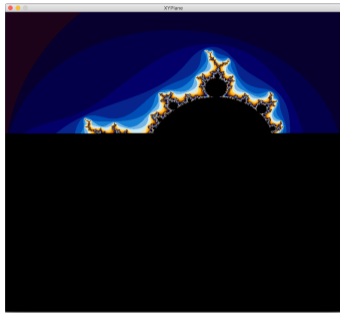
in Java or C++

Seq vs Par
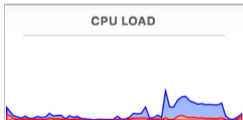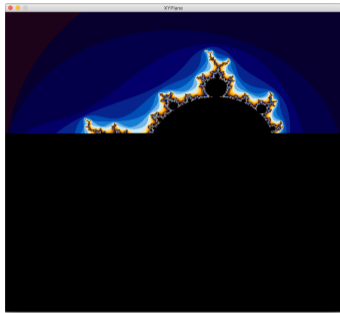
In FP: Once a variable is created, it is assigned a value and then never changed again $\Rightarrow$ no synchronisation needed

Types

- Base types
  Int, Long, BigInt, Float, Double
  String, Char
  Boolean

- Compound types

List[Int]	lists of Int's
Set[Double]	sets of Double's
(Int, String)	Int-String pair
List[(BigInt, String)]	lists of BigInt-String pairs
List[List[Int]]	list of lists of Int's
Option[Int]	options of Int's

```scala
def fname(arg1: ty1, arg2: ty2,..., argn: tyn): rty = {

    ....

}
```

```scala
def average(xs: List[Int]) : Int = {
  val s = xs.sum
  val n = xs.length
  s / n
}
```

The Joy of Immutability

- If you need to manipulate some data in a list say, then you make a new list with the updated values, rather than revise the original list. Easy!

```scala
val old_list = List(1, 2, 3, 5)
val new_list = 0 :: old_list
          // -> List(0, 1, 2, 3, 4, 5)
```

- You do not have to be defensive about who can access the data.
- You can look at your code in isolation.

Email: Hate 'val'

Subject: **Hate 'val'**   01:00 AM

Hello Mr Urban,

I just wanted to ask, how are we suppose to work with the completely useless **val**, that can't be changed ever? Why is this rule active at all? I've spent 4 hours not thinking on the coursework, but how to bypass this annoying rule. What's the whole point of all these coursework, when we can't use everything Scala gives us?!?

Regards.

« deleted »

Subject: **Re: Hate 'val'**   01:02 AM

« my usual rant about fp...
concurrency bla bla... better programs yada »

PS: What are you trying to do where you desperately
want to use `var`?

Subject: **Re: Re: Hate 'val'**   01:04 AM

Right now my is_legal function works fine:

```scala
def is_legal(dim: Int, path: Path)(x: Pos): Boolean = {
  var boolReturn = false
  if(x._1 > dim || x._2 > dim || x._1 < 0 || x._2 < 0) {
  else { var breakLoop = false
        if(path == Nil) { boolReturn = true }
        else { for(i <- 0 until path.length) {
                 if(breakLoop == false) {
                   if(path(i) == x) {
                     boolReturn = true
                     breakLoop = true
                   }
                   else { boolReturn = false }
                 } else breakLoop
               }
        }
        boolReturn
  }
```

...but I can't make it work with boolReturn being val. What approach would you recommend in this case, and is using var in this case justified?

Right now my is_legal function works fine:

```scala
def is_legal(dim: Int, path: Path)(x: Pos): Boolean = {
  var boolReturn = false
  if(x._1 > dim || x._2 > dim || x._1 < 0 || x._2 < 0) {
  else { var breakLoop = false
          if(path == Nil) { boolReturn = true }
          else { for(i <- 0 until path.length) {
                    if(breakLoop == false) {
                      if(path(i) == x) {
                        boolReturn = true
                        breakLoop = true
                      }
                      else { boolReturn = false }
                    } else breakLoop
```

Me:

...but I can't make it work with boolReturn being val. What approach would you recommend in this case, and is using var in this case justified?

```
                  turn
```
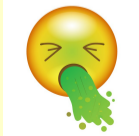
Subject: **Re: Re: Re: Hate 'val'**   01:06 AM

OK. So you want to make sure that the x-position is not outside the
board....and furthermore you want to make sure that the x-position is
not yet in the path list. How about something like

```
def is_legal(dim: Int, path: Path)(x: Pos): Boolean =
  ...<<some board conditions>>... && !path.contains(x)
```

Does not even contain a `val`.

(This is all on one line)

Subject: **Re: Re: Re: Re: Hate 'val'**   11:02 AM

THANK YOU! You made me change my coding
perspective. Because of you, I figured out the next
one...

Subject: **Re: Re: Re: Re: Hate 'val'**   11:02 AM

THANK YOU! You made me change my coding
perspective. Because of you, I figured out the next
one...

Me:

Conclusion for Today

- Scala is still under development, 2.13.1 came out in Sept.
  (the compiler is terribly slow)

- **http://www.scala-lang.org/**

- it is a rather **deep** language...i.e. gives you a lot of rope to shoot yourself

- learning functional programming is not easy...when you have spent all of your career thinking in an imperative way, it is hard to change

- hope you have fun with Scala and the assignments