Scala First:
Lessons from 3 student generations

Bjorn Regnell

Dept. of Computer Science, Lund University

Who is Bjorn Regnell?

- Professor in Software Engineering
- Dept. of Computer Science, Engineering Faculty LTH, Lund University, Sweden
- Research focus: Software Requirements Engineering
  - **`http://cs.lth.se/bjorn-regnell`**
- Scala programmer since 2010
- Teaching Scala using Kojo IDE in primary school to kids and teachers since 2011
  - **`http://www.kogics.net/kojo-download`**
- Teaching Scala at university level since 2016
  - **`https://github.com/lunduniversity/introprog`**

LUND UNIVERSITY

Acknowledgements

Many thanks to

- hard-working students
- enthusiastic teaching assistants
- supportive colleagues
- contributors to open source course material
  - `https://github.com/lunduniversity/introprog/blob/master/contributors.tex`
- the fantastic Scala community

Scala first lessons

Agenda:

- Why Scala first?
- How did we implement Scala first?
- What did we learn?
- The road ahead

Why Scala first?

About our Scala students at Lund Univesity

- Enrolled at the 5 year program in Computer Science & Engineering (CSE) at LTH
- Taking their first programming course 7.5 ECTS credits during their first semester
- More than 80% are 19–22 years old
- Around 15% are female
- Around 65% have programmed before (C#, Java, Python, Javascript, ...)
- Very big span of pre-knowledge
- Increasingly selective admission: 136 accepted out of 1337 applicants (2018)

History of first languages at Lund University

First languages for CSE students at LTH:

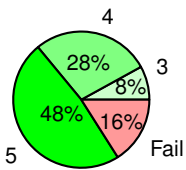
(Algol)	(pre-history using punch cards)
Pascal	1982, CSE program started
Simula	1990
Java	1997
Scala	2016

The Java first situation (2015)

I took over as course head in 2015 and ran it as usual with Java first, while in parallell heading the development of the Scala first course material as an open source community project with colleagues and senior students.

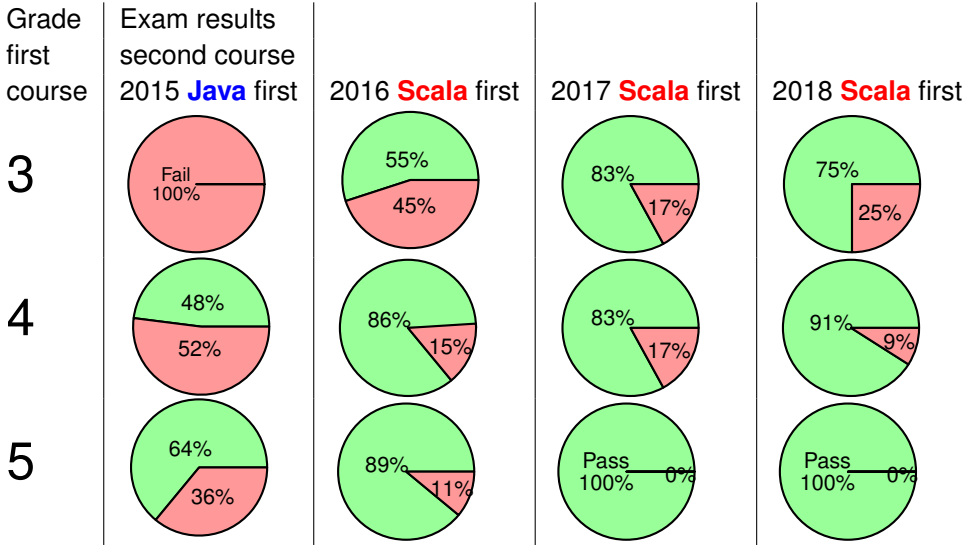
Good:

- Course very popular
- Exam results very good:



Bad:

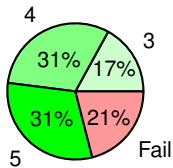
- Competent students not challenged
- Poor results in second course



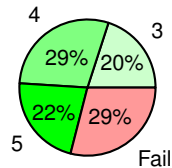
Scala first – a big success! :)

- Exam results in second course much better: increased learning outcome
- Course evaluations are still very positive: increased student engagement
- All students are challenged, independent of pre-knowledge
- Students find Scala interesting and novel
- Exam results:

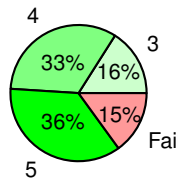
2016



2017



2018



Why not Python first?

My hypotheses based on some experience (but no systematic empiricism):

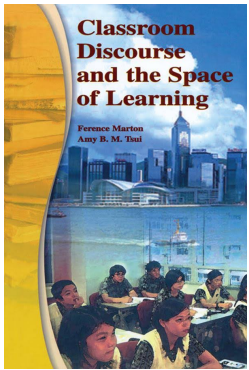
- Non-explicit types in function defs is less efficient when learning abstract thinking
- Indentation syntax with silent begin–end makes nested blocks obscure to beginners
- Dynamic typing means less help in finding bugs; lack of motivation when stuck
- No explicit variable declaration can lead to very hard bugs also for non-beginners
- Not excellent at object orientation
- Not excellent at functional programming

Why Scala first?

My hypotheses based on some experience (but no systematic empiricism):

- Easy for beginners and interesting for non-beginners (depending how you teach...)
- Regular semantics is easier for beginners; e.g. value types are real objects
- Concise (but not too concise), expressive: do interesting things with small programs
- Static typing help finding bugs: the compiler is your friend
- Interactive learning in the Scala REPL: understand expressions step by step
- Modern, evolving language: exciting for students and teachers
- Multi-paradigm: excellent for mixing imperative, object-oriented, functional

Pedagical support for multi-paradigm language



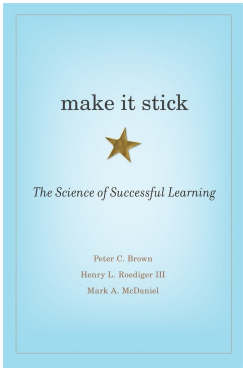
Marton & Tsui (2004)

Patterns of variation

- Patterns of variation

How did we implement Scala first?

Pedagical ideas behind course design



Brown et al. (2014)

What did we learn?

The road ahead

LUND
UNIVERSITY