Functional Programming **for All**!

Scaling a MOOC for Students and Professionals Alike

Heather Miller

TFPIE'17, Canterbury, UK
June 21st, 2017

Northeastern

EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

First of all, this wasn’t all done by me alone.

Others who helped make our MOOC story possible:

Lukas Rytz
Vojin Jovanovic
Manohar Jonnalagedda
Aleksandar Prokopec
Jorge Vicente Cantero
Martin Odersky
Viktor Kuncak
Erik Meijer
Tao Lee
Tobias Schlatter
Philipp Haller
Julien Richard-Foy
Fengyun Liu

Agenda

the courses
tools & infrastructure
the data we collected (it’s open-source!)
our impressions

Agenda

the courses
tools & infrastructure
the data we collected (it's open-source!)
our impressions

My goal in this talk:
To give you as complete of an impression as I can about the full experience of running a popular MOOC on functional programming.

Our foray into MOOCs...

At a glance:

To date, **6** MOOCs

~800,000 learners reached

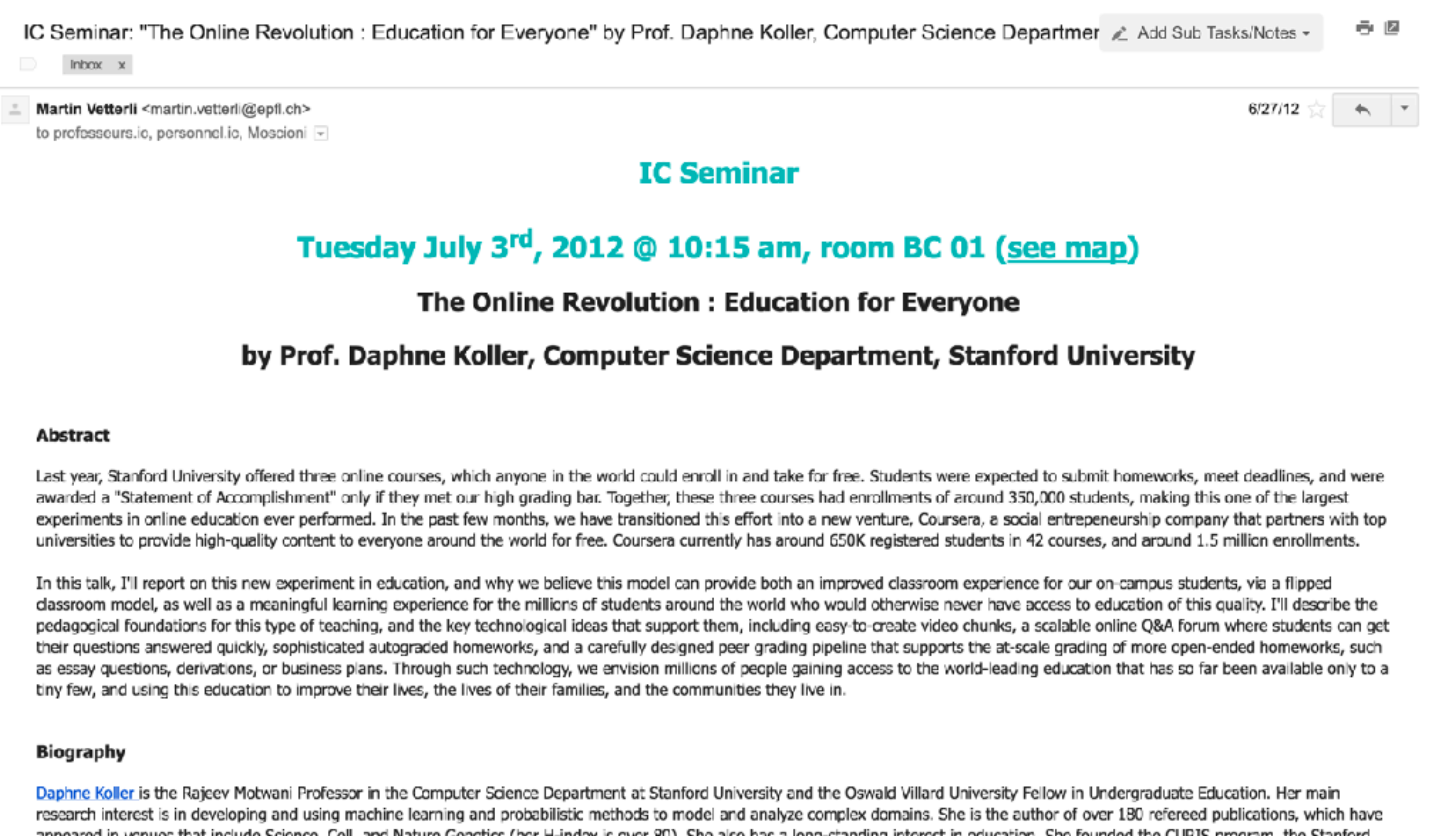
Started in 2012:

Functional Programming Principles in Scala

on: coursera

↑ *(in its infancy at the time)*

Daphne Koller visited us at EPFL in July 2012:



IC Seminar: "The Online Revolution : Education for Everyone" by Prof. Daphne Koller, Computer Science Departmer

Martin Vetterli <martin.vetterli@epfl.ch>
to professeurs.ic, personnel.ic, Moscioni

6/27/12

IC Seminar

Tuesday July 3rd, 2012 @ 10:15 am, room BC 01 (see map)

The Online Revolution : Education for Everyone

by Prof. Daphne Koller, Computer Science Department, Stanford University

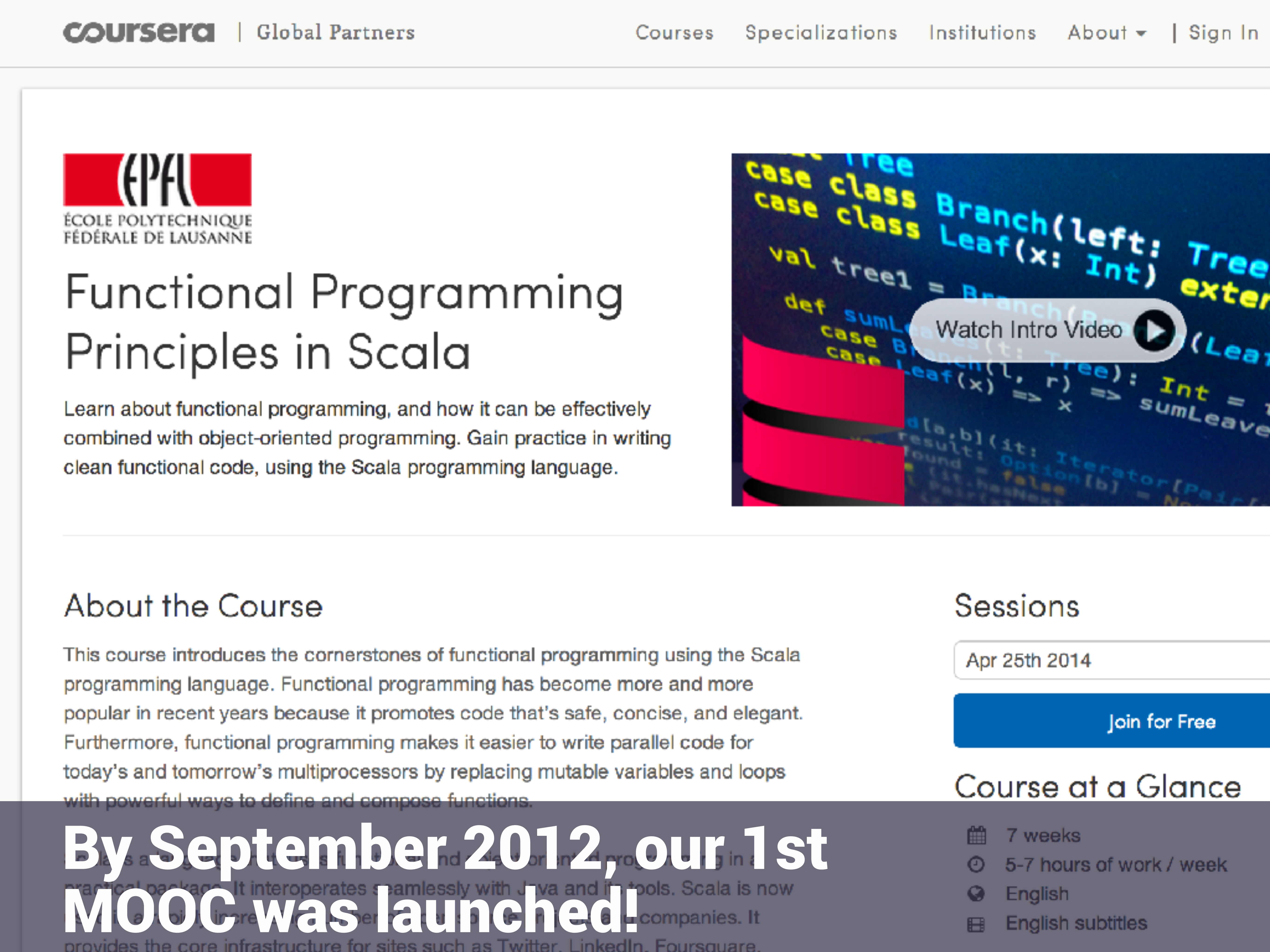
Abstract

Last year, Stanford University offered three online courses, which anyone in the world could enroll in and take for free. Students were expected to submit homeworks, meet deadlines, and were awarded a "Statement of Accomplishment" only if they met our high grading bar. Together, these three courses had enrollments of around 350,000 students, making this one of the largest experiments in online education ever performed. In the past few months, we have transitioned this effort into a new venture, Coursera, a social entrepeneurship company that partners with top universities to provide high-quality content to everyone around the world for free. Coursera currently has around 650K registered students in 42 courses, and around 1.5 million enrollments.

In this talk, I'll report on this new experiment in education, and why we believe this model can provide both an improved classroom experience for our on-campus students, via a flipped classroom model, as well as a meaningful learning experience for the millions of students around the world who would otherwise never have access to education of this quality. I'll describe the pedagogical foundations for this type of teaching, and the key technological ideas that support them, including easy-to-create video chunks, a scalable online Q&A forum where students can get their questions answered quickly, sophisticated autograded homeworks, and a carefully designed peer grading pipeline that supports the at-scale grading of more open-ended homeworks, such as essay questions, derivations, or business plans. Through such technology, we envision millions of people gaining access to the world-leading education that has so far been available only to a tiny few, and using this education to improve their lives, the lives of their families, and the communities they live in.

Biography

Daphne Koller is the Rajeev Motwani Professor in the Computer Science Department at Stanford University and the Oswald Villard University Fellow in Undergraduate Education. Her main research interest is in developing and using machine learning and probabilistic methods to model and analyze complex domains. She is the author of over 180 refereed publications, which have appeared in venues that include Science, Cell, and Nature Genetics (her H-index is over 80). She also has a long-standing interest in education. She founded the CURIS program, the Stanford



Functional Programming Principles in Scala

Learn about functional programming, and how it can be effectively combined with object-oriented programming. Gain practice in writing clean functional code, using the Scala programming language.

About the Course

This course introduces the cornerstones of functional programming using the Scala programming language. Functional programming has become more and more popular in recent years because it promotes code that's safe, concise, and elegant. Furthermore, functional programming makes it easier to write parallel code for today's and tomorrow's multiprocessors by replacing mutable variables and loops with powerful ways to define and compose functions.

[illegible] in a practical package. It interoperates seamlessly with Java and its tools. Scala is now [illegible] companies. It provides the core infrastructure for sites such as Twitter, LinkedIn, Foursquare,

Sessions

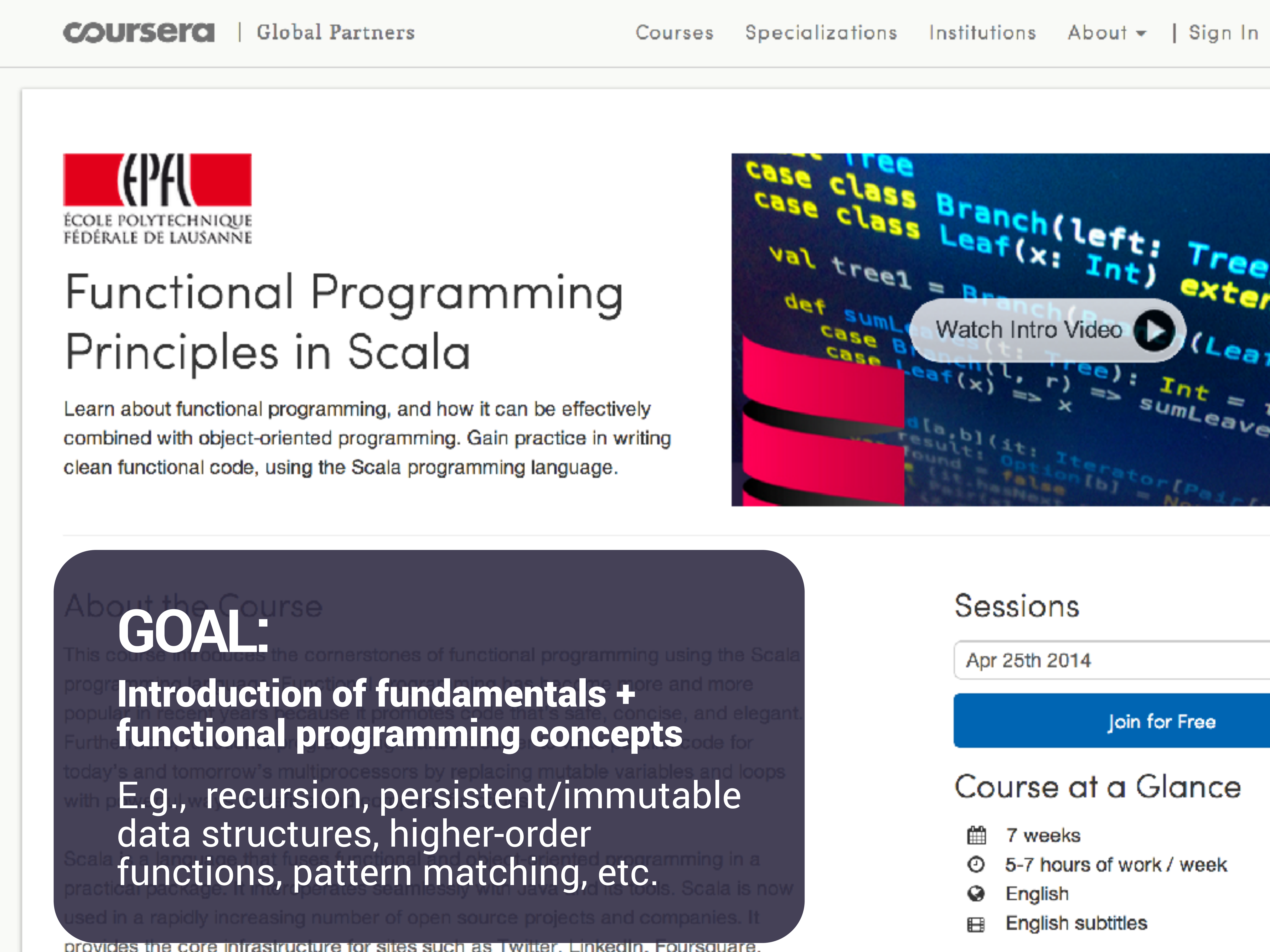
Apr 25th 2014

Join for Free

Course at a Glance

- 7 weeks
- 5-7 hours of work / week
- English
- English subtitles

By September 2012, our 1st MOOC was launched!



Functional Programming Principles in Scala

Learn about functional programming, and how it can be effectively combined with object-oriented programming. Gain practice in writing clean functional code, using the Scala programming language.

Sessions

Apr 25th 2014

Course at a Glance

- 7 weeks
- 5-7 hours of work / week
- English
- English subtitles

GOAL:

Introduction of fundamentals + functional programming concepts

E.g., recursion, persistent/immutable data structures, higher-order functions, pattern matching, etc.

Preliminaries

7 weeks.

- workload: 5-7 hours per week
- verbatim 50% of EPFL's on-campus Functional Programming course (2nd year bachelor level)

Lecture videos.

- each 6-8 minutes long
- total 1.5-2 hours per week

In-video quizzes.

Auto-graded programming assignments.

Content:

week 1: functions & evaluation, recursion

week 2: higher-order functions

week 3: data and abstraction

week 4: types and pattern matching

week 5: functional lists

week 6: list comprehensions + maps

week 7: streams & lazy evaluation

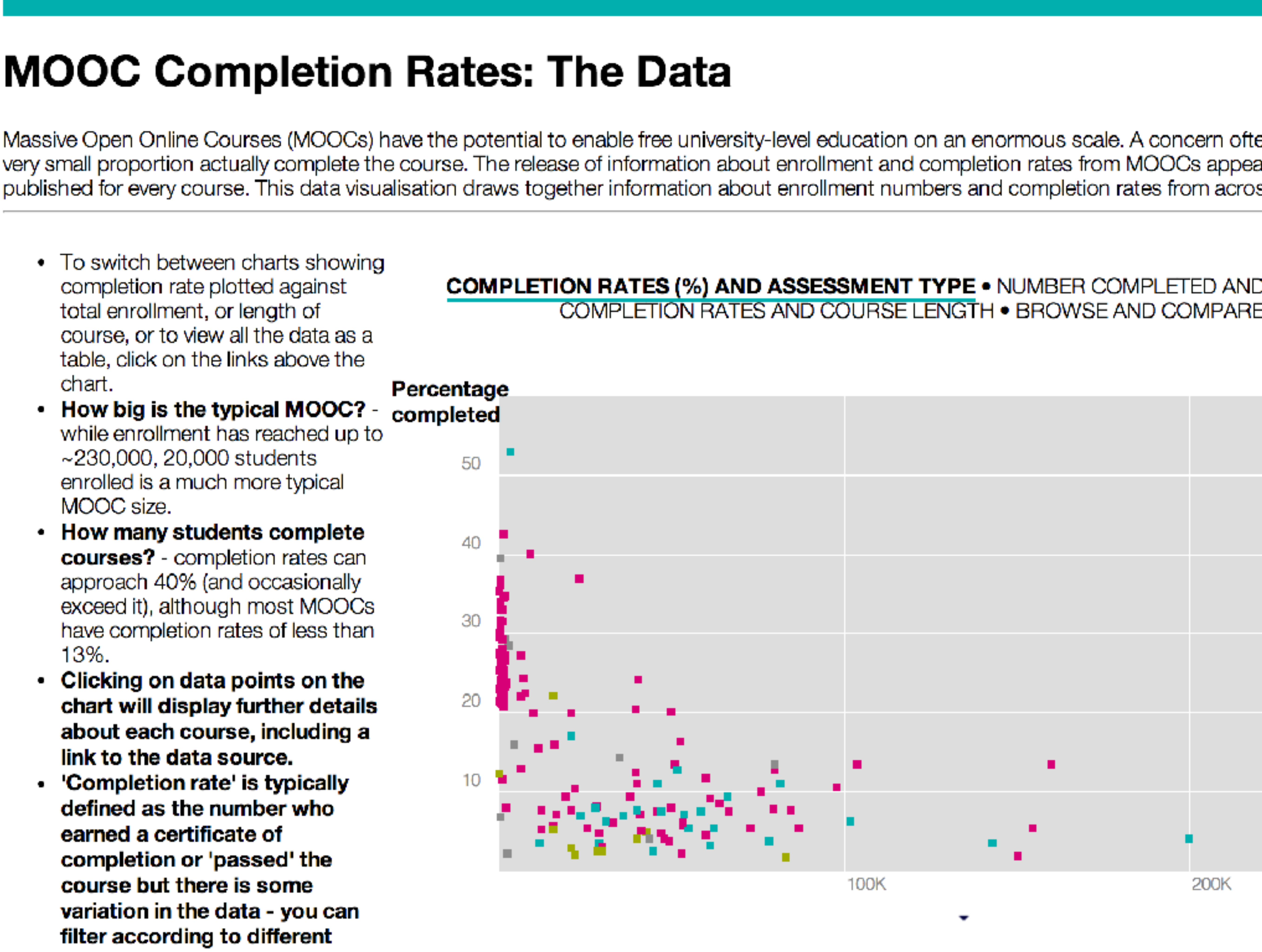
Taught by:
Martin Odersky

Content:

week 1: functions & evaluation, recursion
week 2: higher-order functions
week 3: data and abstraction
week 4: types and pattern matching
week 5: functional lists
week 6: list comprehensions + maps
week 7: streams & lazy evaluation

Taught by:
Martin Odersky

Ok. How'd it go?

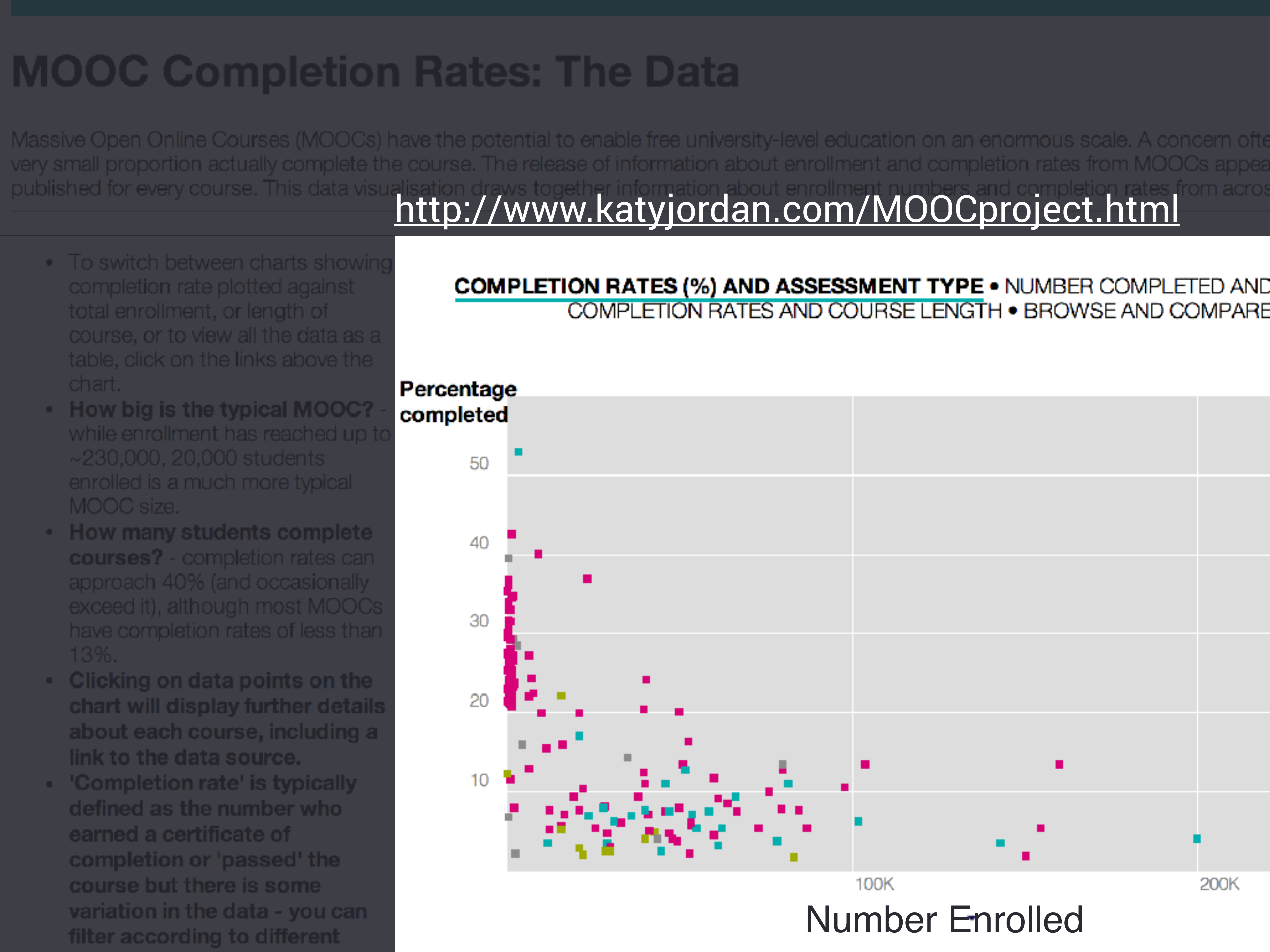


MOOC Completion Rates: The Data

Massive Open Online Courses (MOOCs) have the potential to enable free university-level education on an enormous scale. A concern ofte
very small proportion actually complete the course. The release of information about enrollment and completion rates from MOOCs appea
published for every course. This data visualisation draws together information about enrollment numbers and completion rates from acros

- To switch between charts showing completion rate plotted against total enrollment, or length of course, or to view all the data as a table, click on the links above the chart.
- **How big is the typical MOOC?** - while enrollment has reached up to ~230,000, 20,000 students enrolled is a much more typical MOOC size.
- **How many students complete courses?** - completion rates can approach 40% (and occasionally exceed it), although most MOOCs have completion rates of less than 13%.
- **Clicking on data points on the chart will display further details about each course, including a link to the data source.**
- **'Completion rate' is typically defined as the number who earned a certificate of completion or 'passed' the course but there is some variation in the data - you can filter according to different**

COMPLETION RATES (%) AND ASSESSMENT TYPE • NUMBER COMPLETED AND
COMPLETION RATES AND COURSE LENGTH • BROWSE AND COMPARE
Percentage completed
50
40
30
20
10
100K
200K



MOOC Completion Rates: The Data

Massive Open Online Courses (MOOCs) have the potential to enable free university-level education on an enormous scale. A concern ofte very small proportion actually complete the course. The release of information about enrollment and completion rates from MOOCs appea published for every course. This data visualisation draws together information about enrollment numbers and completion rates from acros

http://www.katyjordan.com/MOOCproject.html

- To switch between charts showing completion rate plotted against total enrollment, or length of course, or to view all the data as a table, click on the links above the chart.
- **How big is the typical MOOC?** - while enrollment has reached up to ~230,000, 20,000 students enrolled is a much more typical MOOC size.
- **How many students complete courses?** - completion rates can approach 40% (and occasionally exceed it), although most MOOCs have completion rates of less than 13%.
- **Clicking on data points on the chart will display further details about each course, including a link to the data source.**
- **'Completion rate' is typically defined as the number who earned a certificate of completion or 'passed' the course but there is some variation in the data - you can filter according to different**

COMPLETION RATES (%) AND ASSESSMENT TYPE • NUMBER COMPLETED AND
COMPLETION RATES AND COURSE LENGTH • BROWSE AND COMPARE
Percentage completed
50
40
30
20
10
100K
200K
Number Enrolled

MOOC Completion Rates: The Data

Massive Open Online Courses (MOOCs) have the potential to enable free university-level education on an enormous scale. A concern ofte
very small proportion actually complete the course. The release of information about enrollment and completion rates from MOOCs appea
published for every course. This data visualisation draws together information about enrollment numbers and completion rates from acros

http://www.katyjordan.com/MOOCproject.html

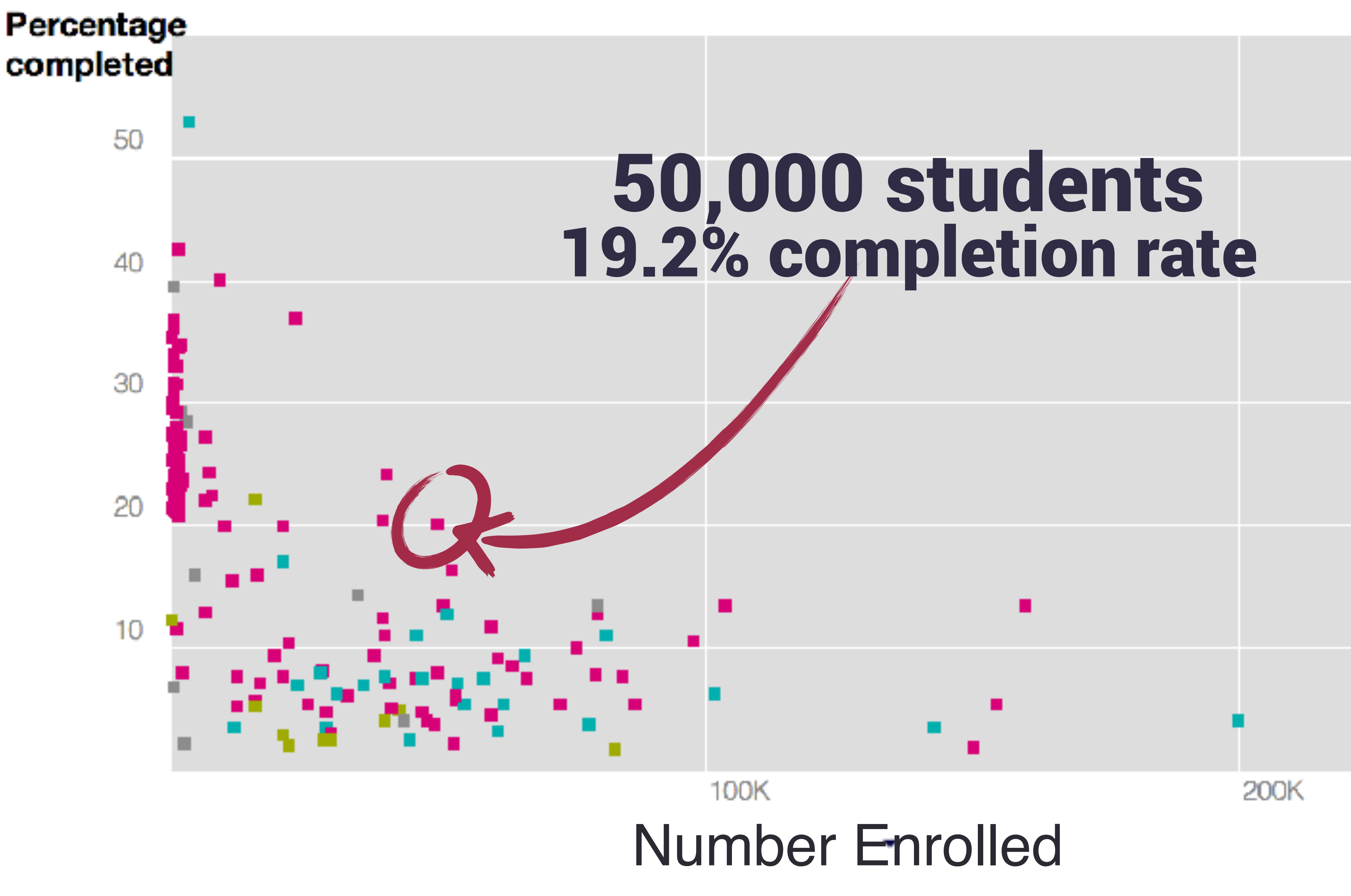
- To switch between charts showing completion rate plotted against total enrollment, or length of course, or to view all the data as a table, click on the links above the chart.
- **How big is the typical MOOC?** - while enrollment has reached up to ~200,000, 20,000 students [illegible] enrollments [illegible] typical MOOC size.
- **How many students complete courses?** - completion rates can approach 40% (and occasionally exceed it), although most MOOCs have completion rates of less than 13%.
- **Clicking on data points on the chart will display further details about each course, including a link to the data source.**
- **'Completion rate' is typically defined as the number who earned a certificate of completion or 'passed' the course but there is some variation in the data - you can filter according to different**

AVERAGE:

across all MOOCs

6.5% completion rate

COMPLETION RATES (%) AND ASSESSMENT TYPE • NUMBER COMPLETED AND
COMPLETION RATES AND COURSE LENGTH • BROWSE AND COMPARE



50,000 students
19.2% completion rate

MOOC Completion Rates: The Data

Massive Open Online Courses (MOOCs) have the potential to enable free university-level education on an enormous scale. A concern ofte
very small proportion actually complete the course. The release of information about enrollment and completion rates from MOOCs appea
published for every course. This data visualisation draws together information about enrollment numbers and completion rates from acros

http://www.katyjordan.com/MOOCproject.html

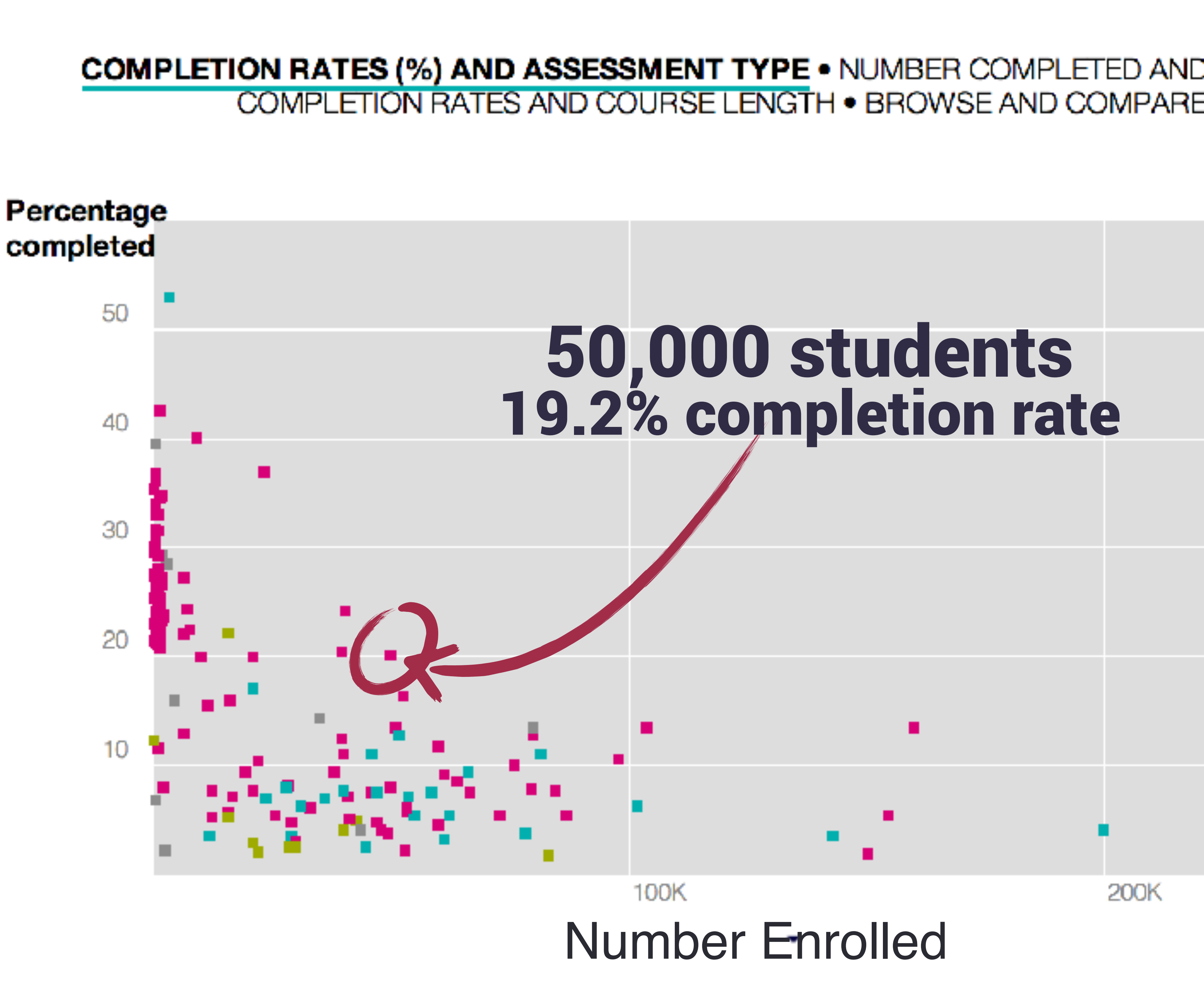
- To switch between charts showing completion rate plotted against total enrollment, or length of course, or to view all the data as a table, click on the links above the chart.
- **How big is the typical MOOC?** - while enrollment has reached up to ~200,000 students [illegible] typical MOOC size.
- **How many students complete courses?** - completion rates can [illegible] exceed it), although most MOOCs have completion rates of less than 10%.
- **Clicking on data points on the chart will display further details about each course, including a** [illegible] **Completion rate is typically defined as the number who earned a certificate of completion or passed the** [illegible] **in the data - you can filter according to different**

AVERAGE:

across all MOOCs

6.5% completion rate

Jordan, K. (2014)
Initial trends in enrollment and completion of massive open online courses.
The International Review of Research in Open and Distance Learning, 15(1), 133-160.



50,000 students
19.2% completion rate

Why such a high completion rate?

our completion rate was 3x the norm.

We think it was the tooling & infrastructure.

automated cloud-based graders

interactive build tool

decent choice of IDEs

style checkers

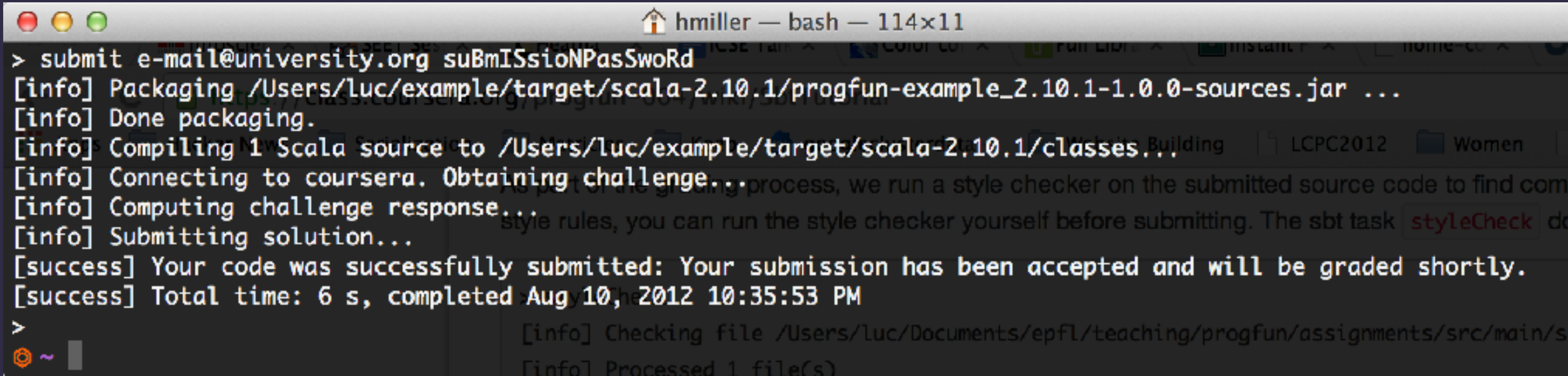
testing frameworks

Interactive development/ submission cycle.

sbt

Compile. Test. Submit.

Scala's interactive build tool, configured to submit student assignments to the automated cloud-based graders from the command line.



```
> submit e-mail@university.org suBmISsioNPasSwoRd
[info] Packaging /Users/luc/example/target/scala-2.10.1/progfun-example_2.10.1-1.0.0-sources.jar ...
[info] Done packaging.
[info] Compiling 1 Scala source to /Users/luc/example/target/scala-2.10.1/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
>
~
```

Automated grading/feedback.

Custom cloud-based auto-grader.

Automated grading/feedback.

Custom cloud-based auto-grader.

Provided two types of feedback:

- ✓ **Massive suite of secret unit tests.**
- ✓ **Style-checker**

Automated grading/feedback.

Custom cloud-based auto-grader.

Provided two types of feedback:

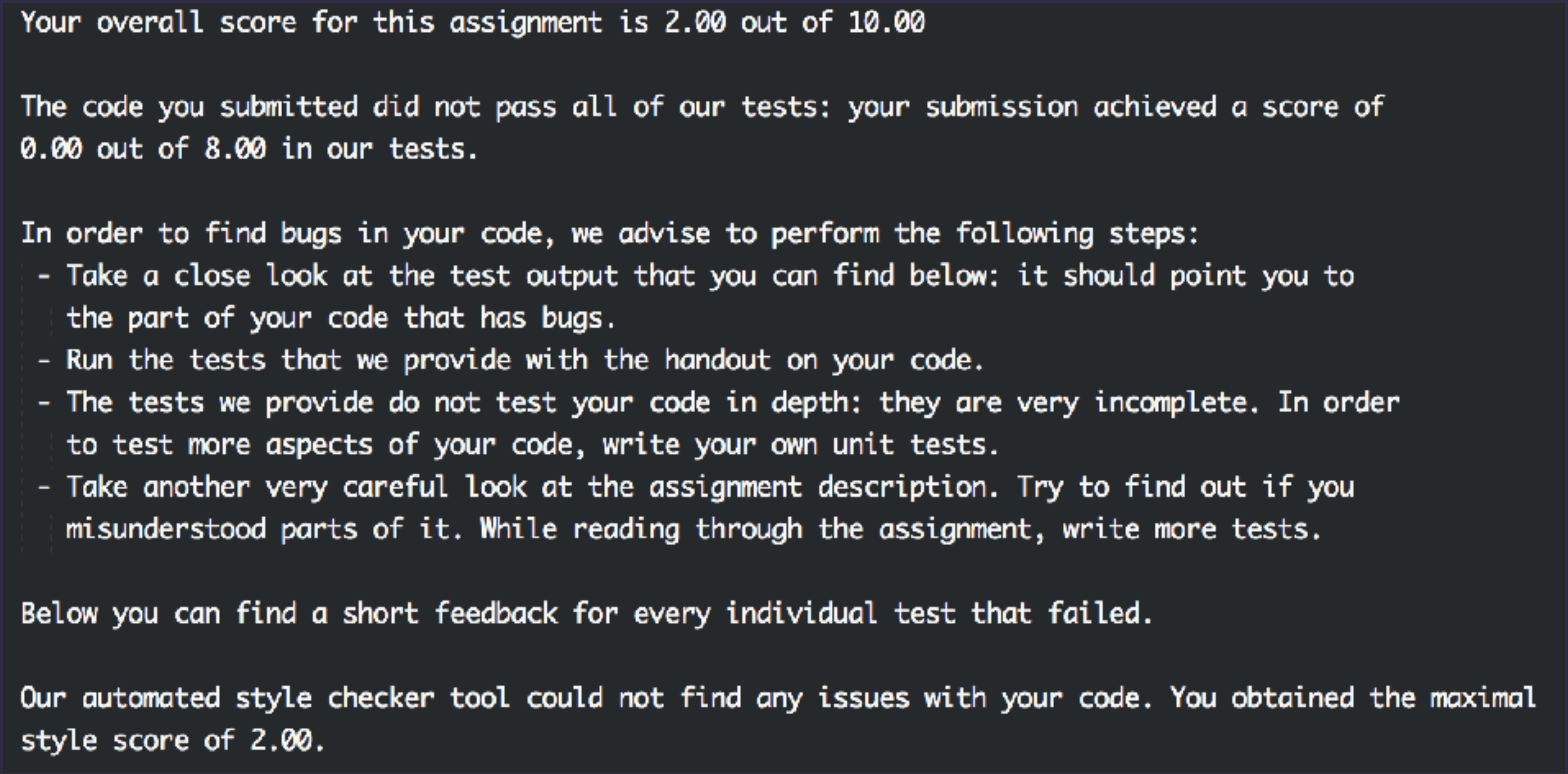
- ✓ **Massive suite of secret unit tests.**
- ✓ **Style-checker**

  discourages:

  – mutable variables
  – return statements
  – the null value
  – while loops
  – magic numbers
  – overly long lines of code
  – non-standard capitalization
  – + more

Automated grading/feedback.

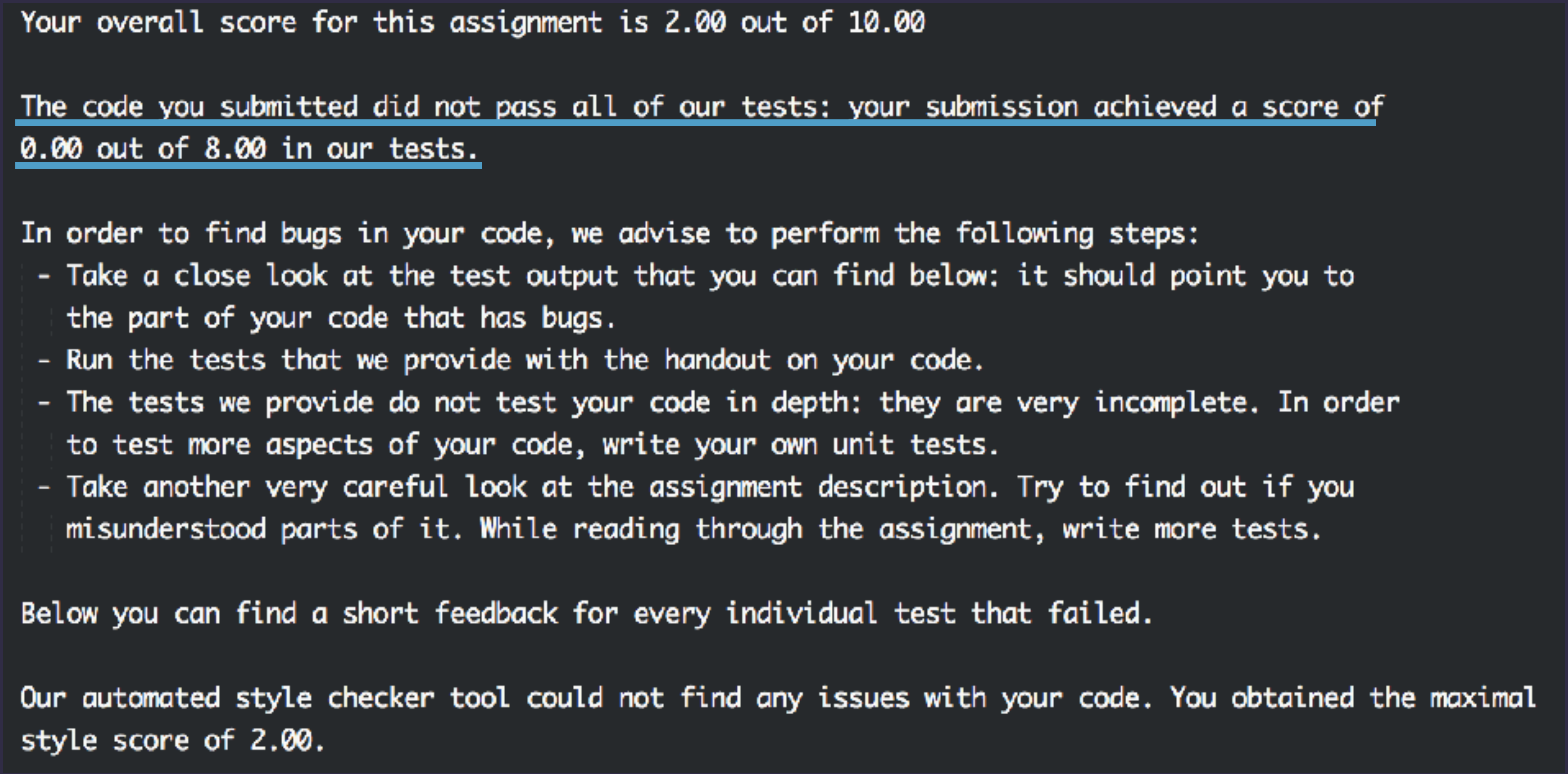
Custom cloud-based auto-grader.



```
Your overall score for this assignment is 2.00 out of 10.00

The code you submitted did not pass all of our tests: your submission achieved a score of
0.00 out of 8.00 in our tests.

In order to find bugs in your code, we advise to perform the following steps:
 - Take a close look at the test output that you can find below: it should point you to
   the part of your code that has bugs.
 - Run the tests that we provide with the handout on your code.
 - The tests we provide do not test your code in depth: they are very incomplete. In order
   to test more aspects of your code, write your own unit tests.
 - Take another very careful look at the assignment description. Try to find out if you
   misunderstood parts of it. While reading through the assignment, write more tests.

Below you can find a short feedback for every individual test that failed.

Our automated style checker tool could not find any issues with your code. You obtained the maximal
style score of 2.00.
```

Automated grading/feedback.

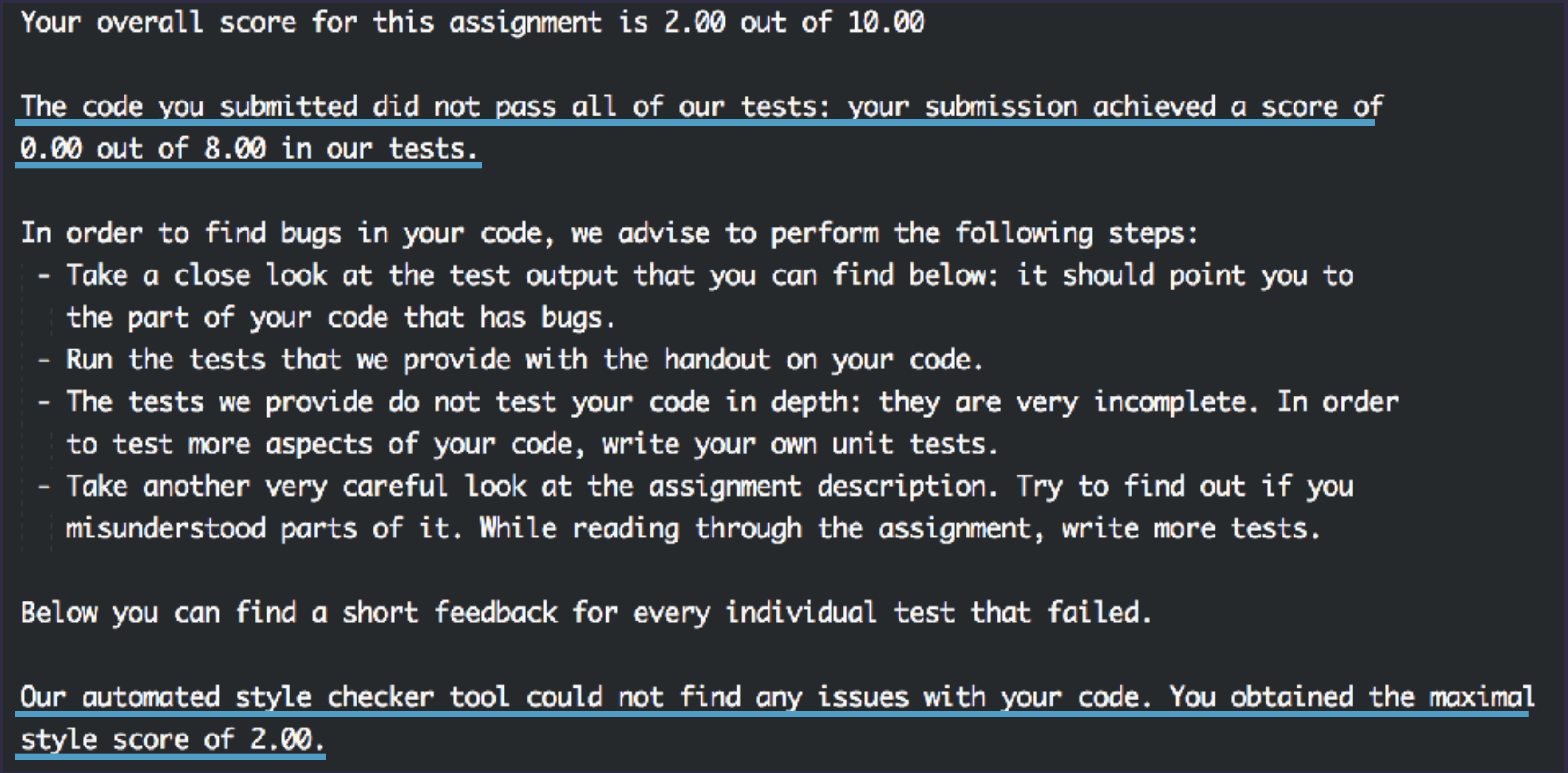
Custom cloud-based auto-grader.



```
Your overall score for this assignment is 2.00 out of 10.00

The code you submitted did not pass all of our tests: your submission achieved a score of
0.00 out of 8.00 in our tests.

In order to find bugs in your code, we advise to perform the following steps:
 - Take a close look at the test output that you can find below: it should point you to
   the part of your code that has bugs.
 - Run the tests that we provide with the handout on your code.
 - The tests we provide do not test your code in depth: they are very incomplete. In order
   to test more aspects of your code, write your own unit tests.
 - Take another very careful look at the assignment description. Try to find out if you
   misunderstood parts of it. While reading through the assignment, write more tests.

Below you can find a short feedback for every individual test that failed.

Our automated style checker tool could not find any issues with your code. You obtained the maximal
style score of 2.00.
```

Automated grading/feedback.

Custom cloud-based auto-grader.



```
Your overall score for this assignment is 2.00 out of 10.00

The code you submitted did not pass all of our tests: your submission achieved a score of
0.00 out of 8.00 in our tests.

In order to find bugs in your code, we advise to perform the following steps:
 - Take a close look at the test output that you can find below: it should point you to
   the part of your code that has bugs.
 - Run the tests that we provide with the handout on your code.
 - The tests we provide do not test your code in depth: they are very incomplete. In order
   to test more aspects of your code, write your own unit tests.
 - Take another very careful look at the assignment description. Try to find out if you
   misunderstood parts of it. While reading through the assignment, write more tests.

Below you can find a short feedback for every individual test that failed.

Our automated style checker tool could not find any issues with your code. You obtained the maximal
style score of 2.00.
```

Automated grading/feedback.

Custom cloud-based auto-grader.

Provided two types of feedback:

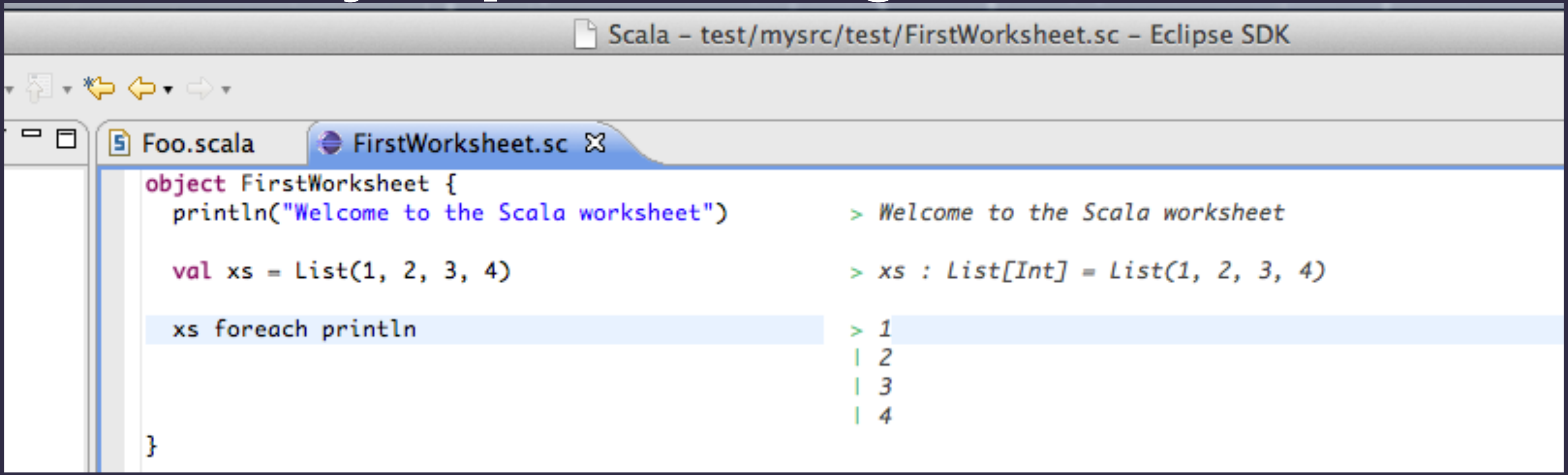
- ✓ **Massive suite of secret unit tests.**
- ✓ **Style-checker**

Importantly:

- ✓ Resubmissions welcome.
- ✓ Feedback arrives fast.
  seconds – 15 minutes

IDEs

Popular IDEs come with worksheets for easy experimenting:

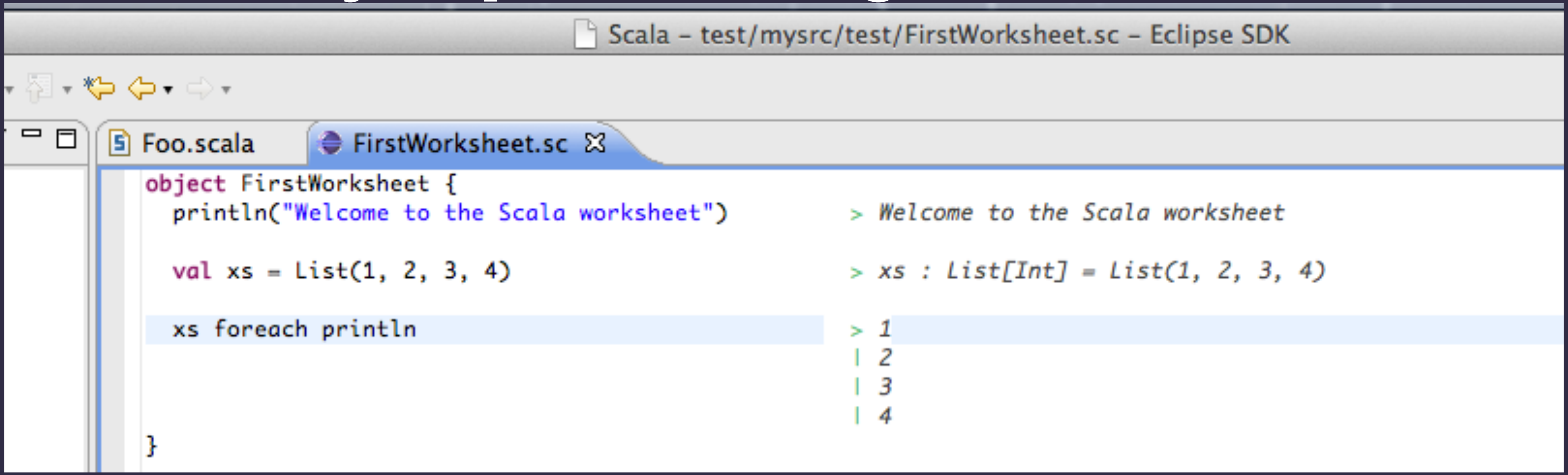


```
object FirstWorksheet {
  println("Welcome to the Scala worksheet")        > Welcome to the Scala worksheet

  val xs = List(1, 2, 3, 4)                         > xs : List[Int] = List(1, 2, 3, 4)

  xs foreach println                                > 1
                                                    | 2
                                                    | 3
                                                    | 4
}
```

IDEs

ScalaIDE
FOR ECLIPSE

Popular IDEs come with worksheets for easy experimenting:



Scala – test/mysrc/test/FirstWorksheet.sc – Eclipse SDK

```
object FirstWorksheet {
  println("Welcome to the Scala worksheet")         > Welcome to the Scala worksheet

  val xs = List(1, 2, 3, 4)                          > xs : List[Int] = List(1, 2, 3, 4)

  xs foreach println                                 > 1
                                                     | 2
                                                     | 3
                                                     | 4
}
```

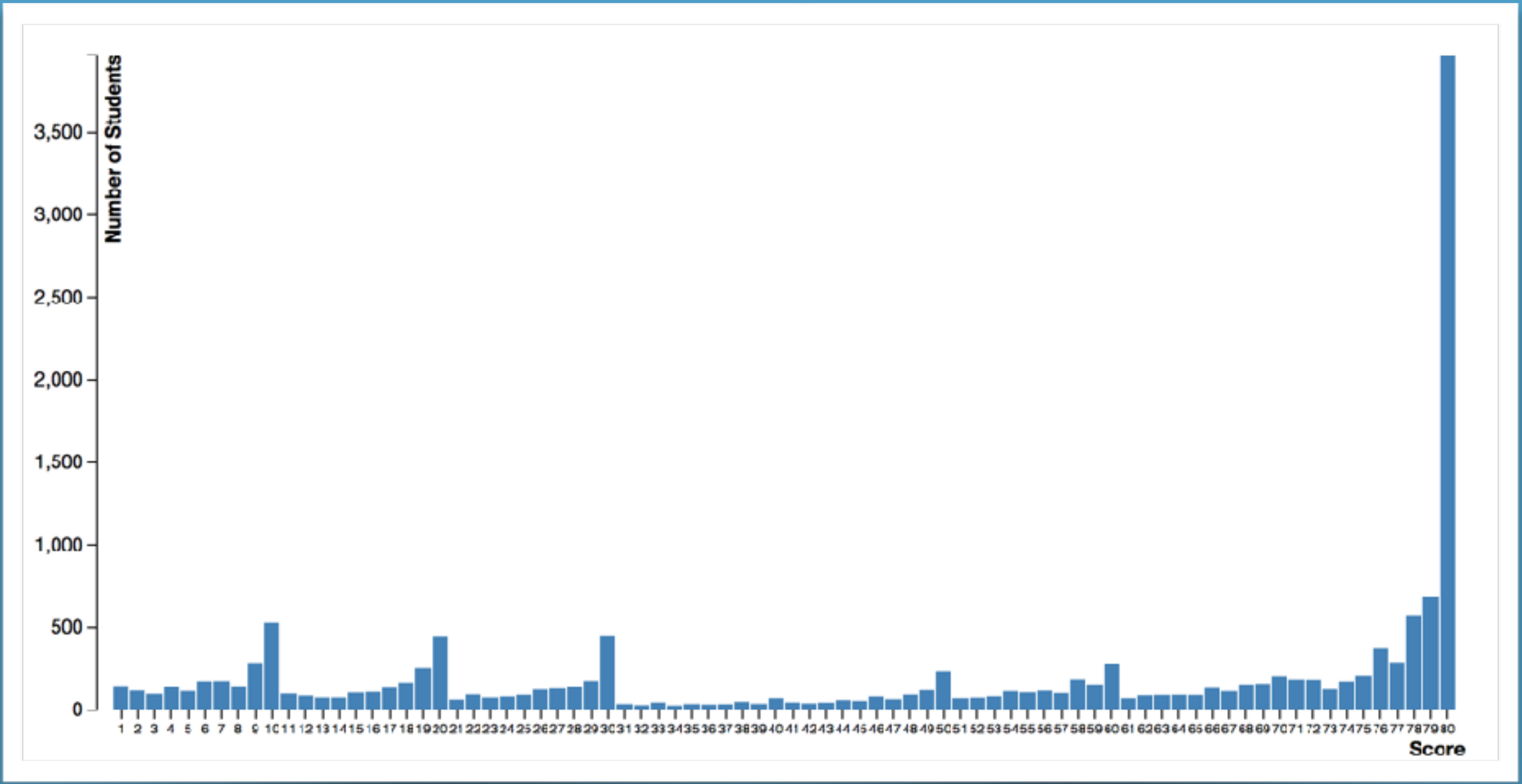
Use sbt right from Eclipse/IntelliJ.
code, compile, test, submit, all from the IDE.

So, what does it mean?

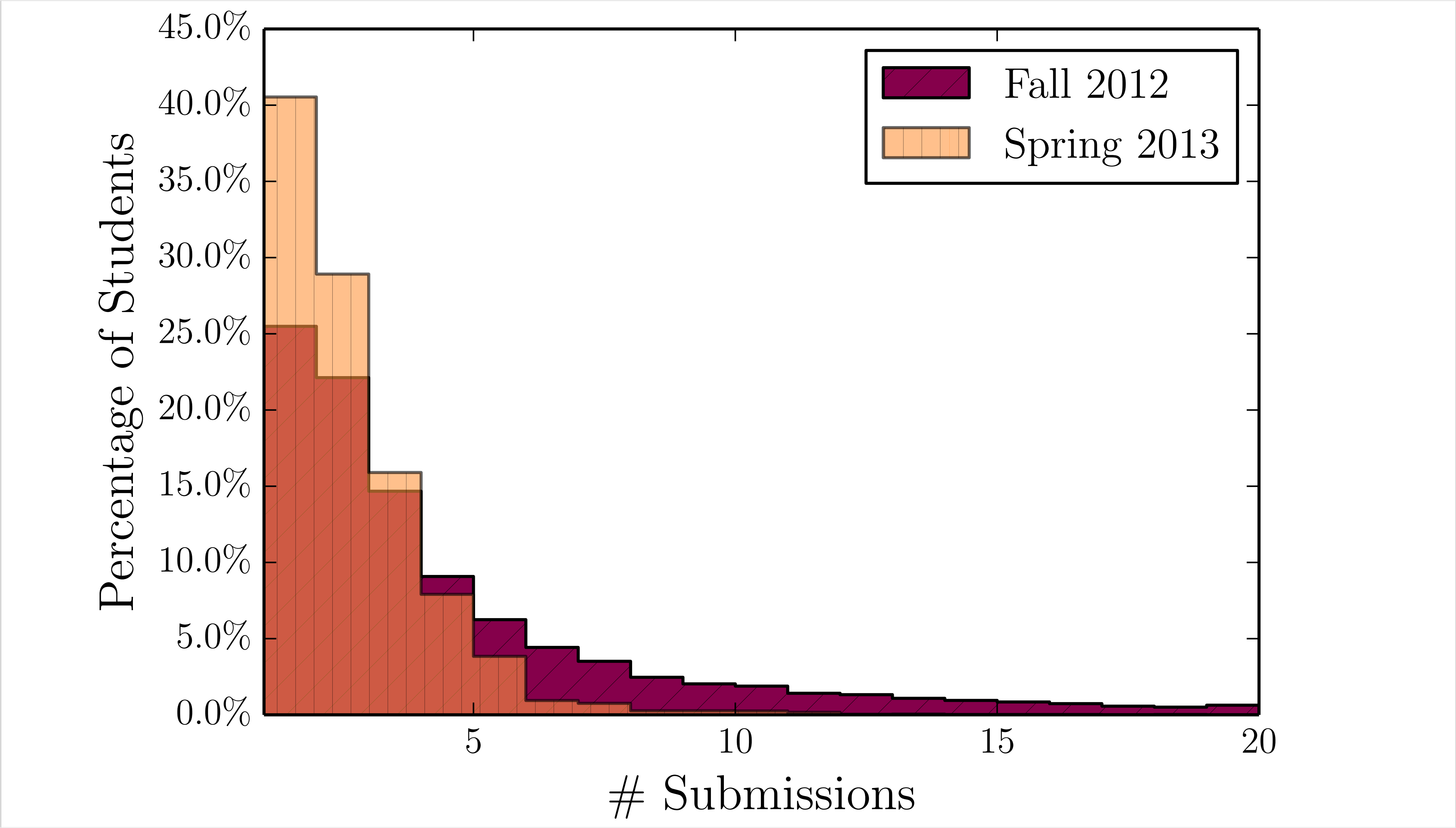
So, what does it mean?

Students had a very tight feedback loop.

If you scored >0, it was most likely that you got 100% (80/80) in the course.



Most people got a perfect score within 4 submission attempts.

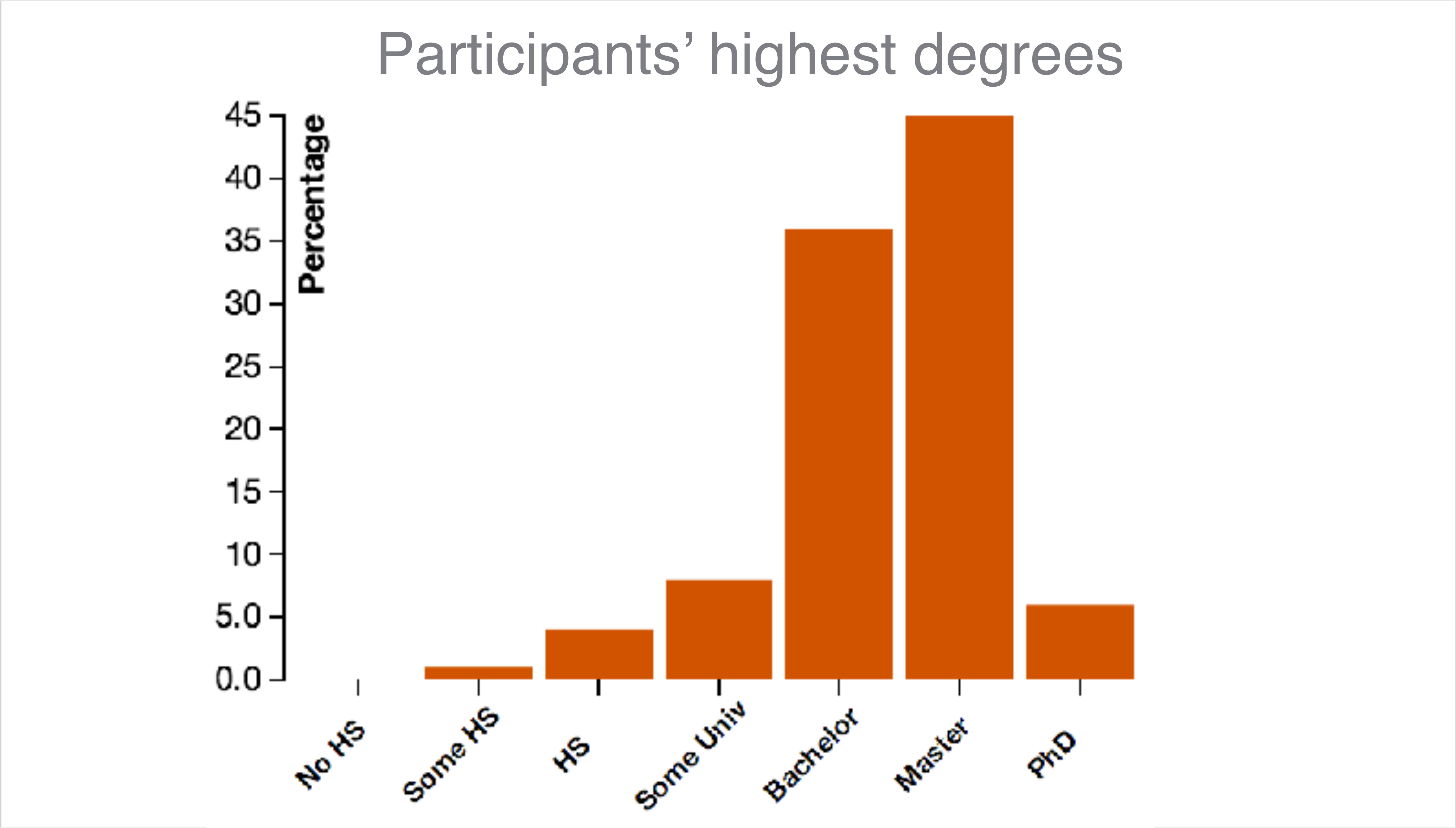


(The number of submissions required to achieve a perfect score.)

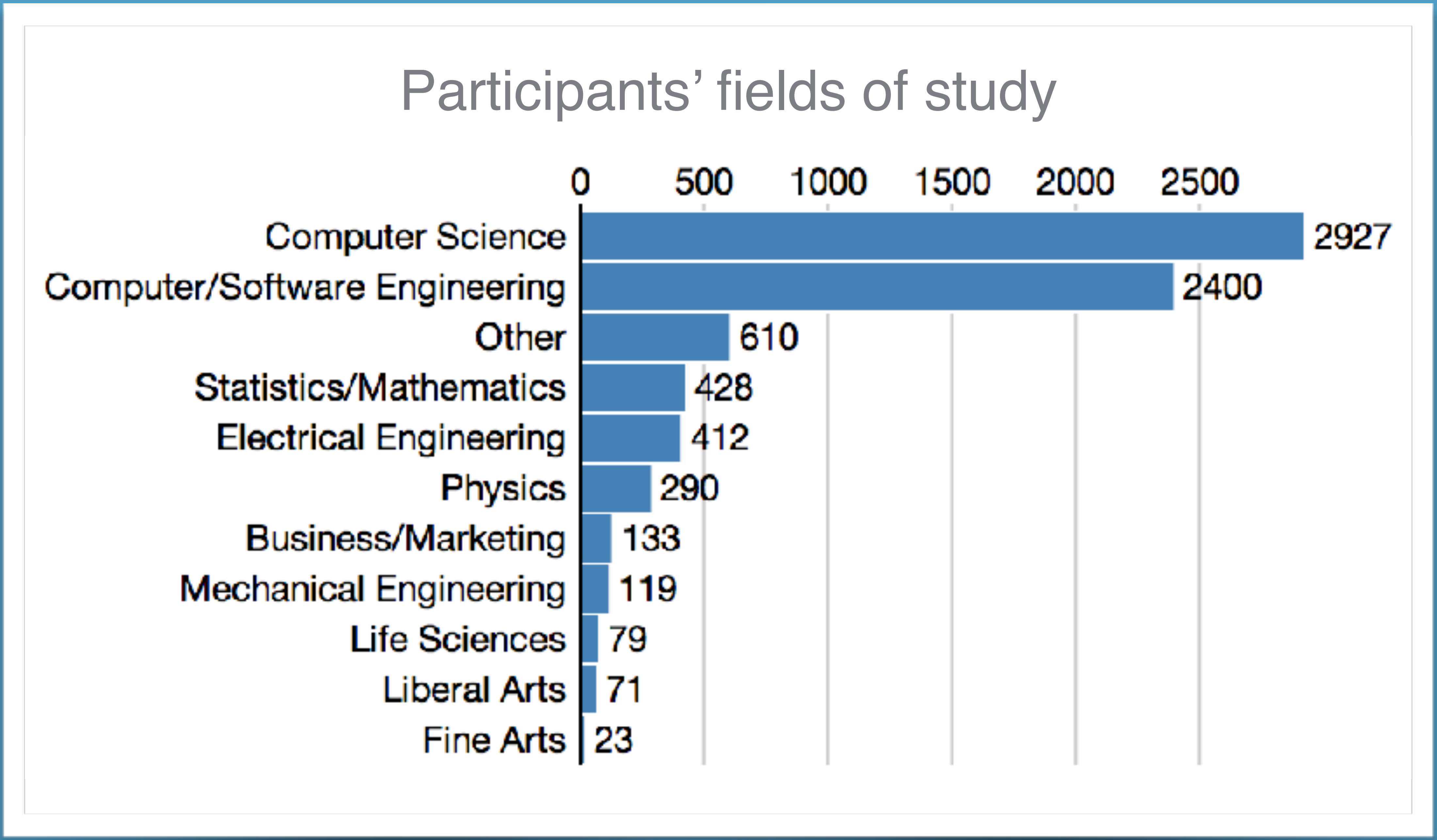
also, this
wasn't just students

but,
professionals too!

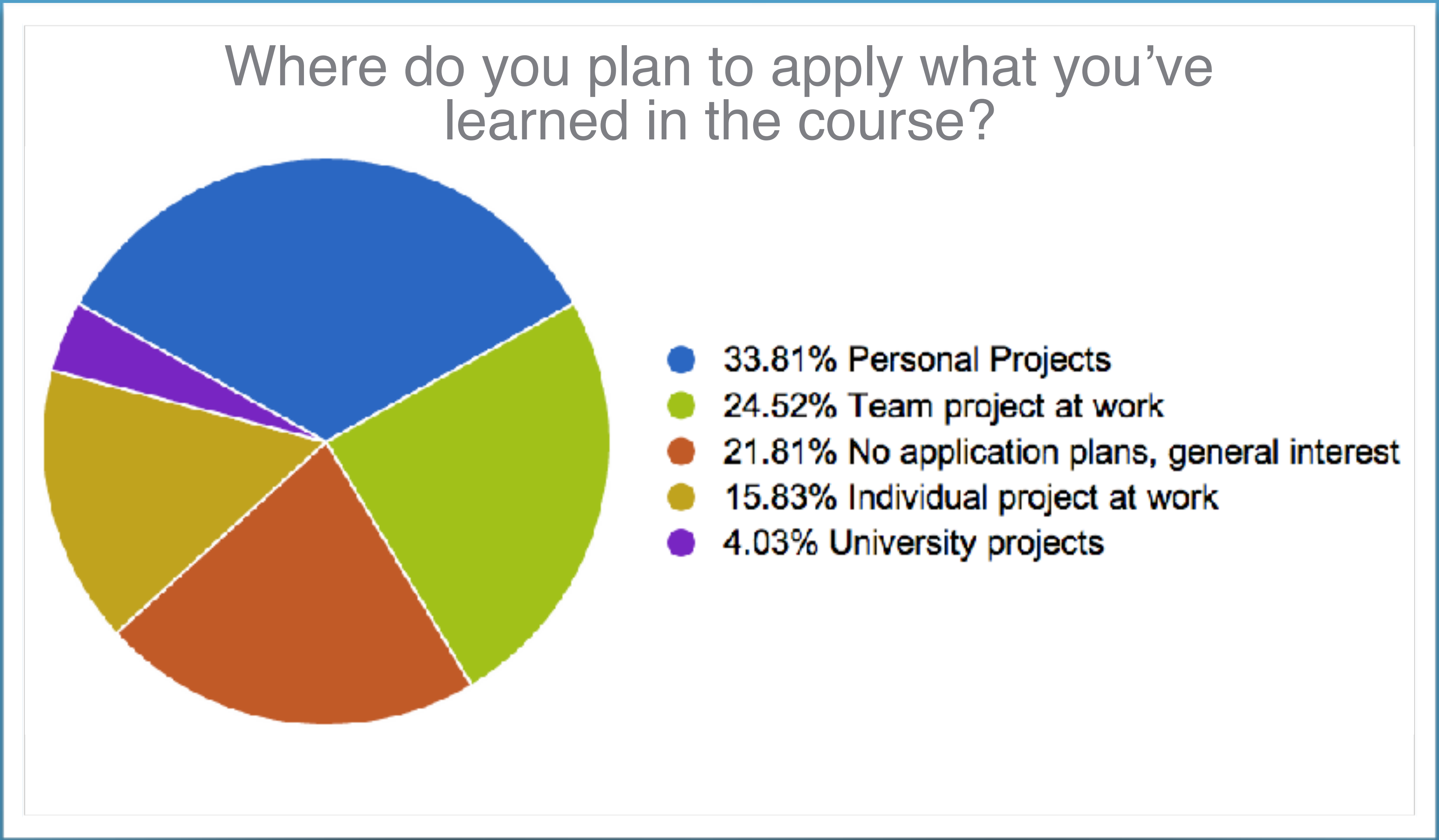
A vast majority of participants already had graduated from university – 87%.



A vast majority of participants come from computer science or computer/software engineering – 71%.

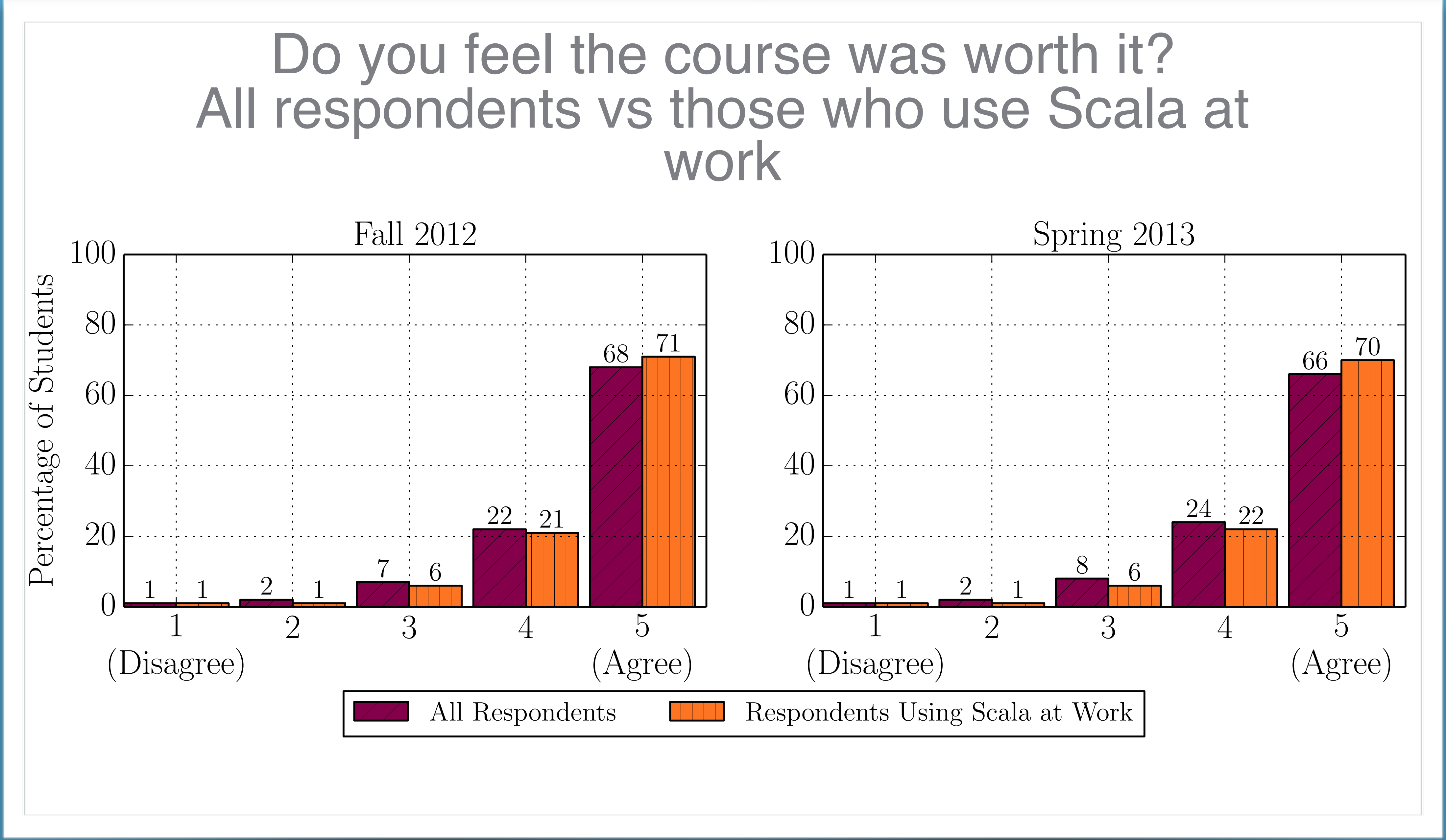


A large portion of participants plan on applying what they’ve learned in the course at work – 40%.



Where do you plan to apply what you’ve learned in the course?

And yet ~70% of professional respondents felt the course was well-worth their time.



For Fall 2012, 71% amounts to 2,148/3,203 professional respondents.

SO,

we can conclude that there were indeed a significant number of professionals participating in the course.

A vast majority of which received perfect scores, and felt that the course was well worth their time.

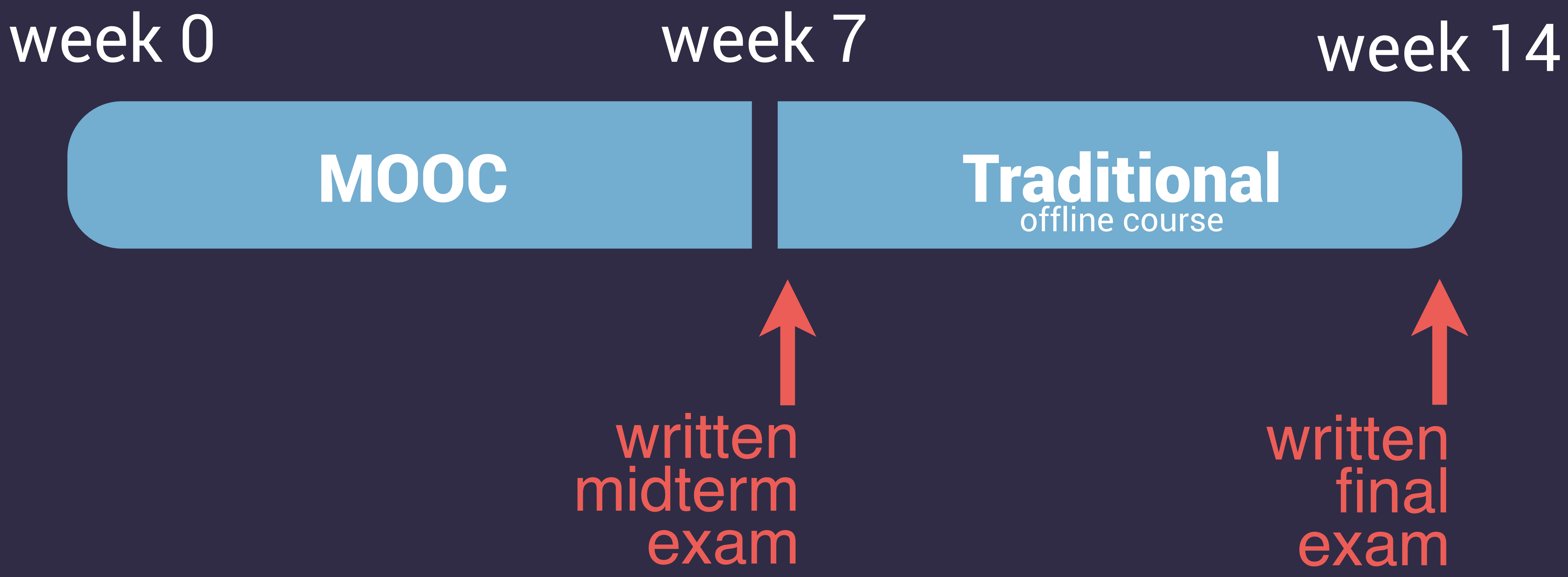
How’d it fare on campus?

Alongside of 50,000 MOOC learners,
150 EPFL students took MOOC for credit.

How’d it fare on campus?

Alongside of 50,000 MOOC learners,
150 EPFL students took MOOC for credit.

EPFL Semester:



week 0

week 7

week 14

MOOC

Traditional
offline course

written
midterm
exam

written
final
exam

How did it differ for EPFL students?

MOOC participants

LECTURES
5-7 videos each week, 8-12min

ASSIGNMENTS
weekly programming exercises

EPFL students

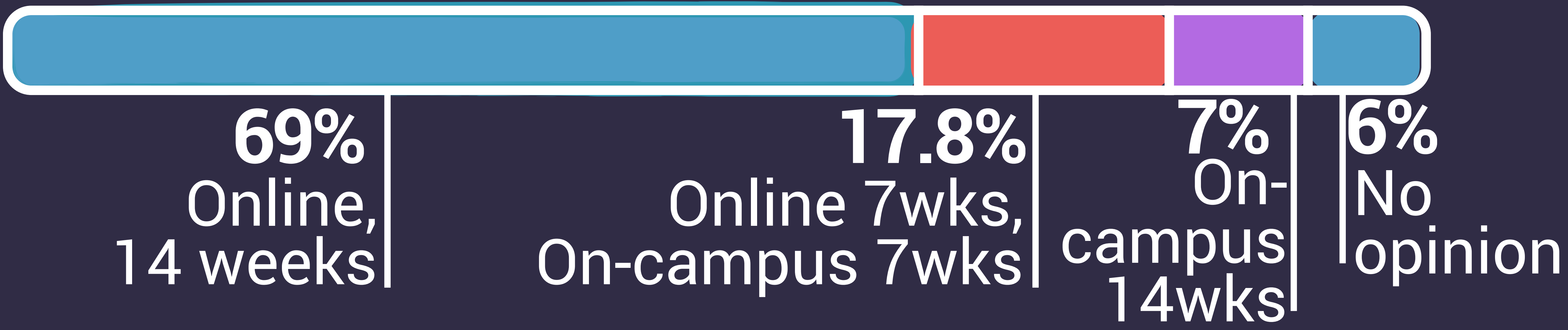
SAME AS MOOC
+
EXERCISE SESSIONS
work in groups, with TAs on HW

WRITTEN EXAMS
midterm & final

offline traditional 2nd half of course

What'd the EPFL students think?

In the future, I'd prefer a course like this be...



Hang on, where did this data come from?

The data

Two iterations of Functional Programming Principles in Scala.

- ✓ Fall 2012
- ✓ Spring 2013

The data

Two iterations of Functional Programming Principles in Scala.

- ✓ Fall 2012
- ✓ Spring 2013

Three sources per iteration:

- Scores & submission data from Coursera
- Survey data
- EPFL specialized course survey

Survey data

Post-course survey:

For the Fall 2012 course, 7,492 respondents out of ~50,000
For the Spring 2013 course, 4,595 respondents out of ~37,000

Total: 12,087 respondents

Survey data

Post-course survey:

For the Fall 2012 course, 7,492 respondents out of ~50,000
For the Spring 2013 course, 4,595 respondents out of ~37,000

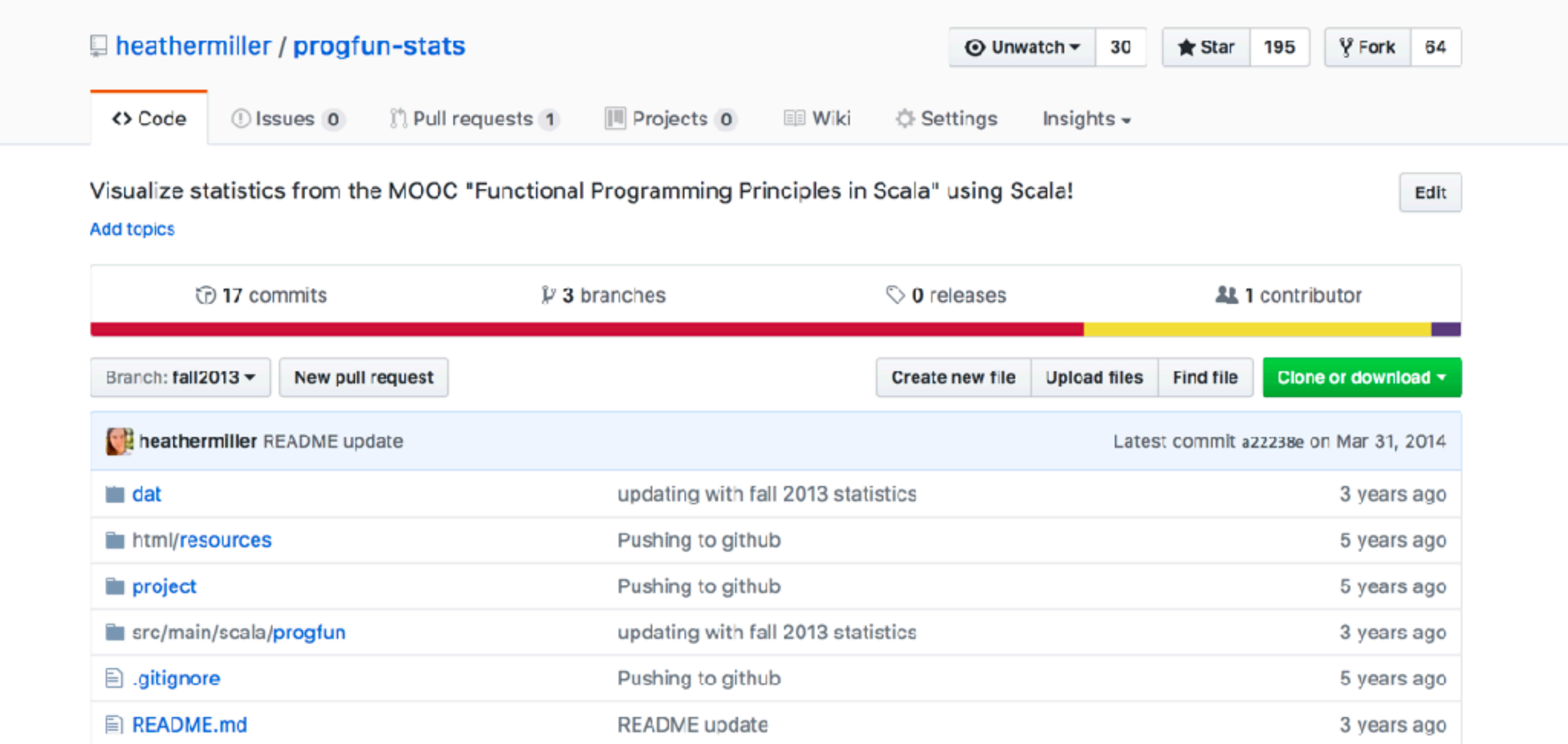
Total: 12,087 respondents

Example questions:

If applicable, what field of study was your highest degree in?
What's your highest degree?
How many years have you been programming?
How difficult did you find the course overall?
Where do you plan to apply what you've learned in this course?
What experience do you have with other programming languages or paradigms?

by the way, the data is open source

+ tools to generate visualizations of the data



Visualize statistics from the MOOC "Functional Programming Principles in Scala" using Scala!

heathermiller README update — Latest commit a22238e on Mar 31, 2014

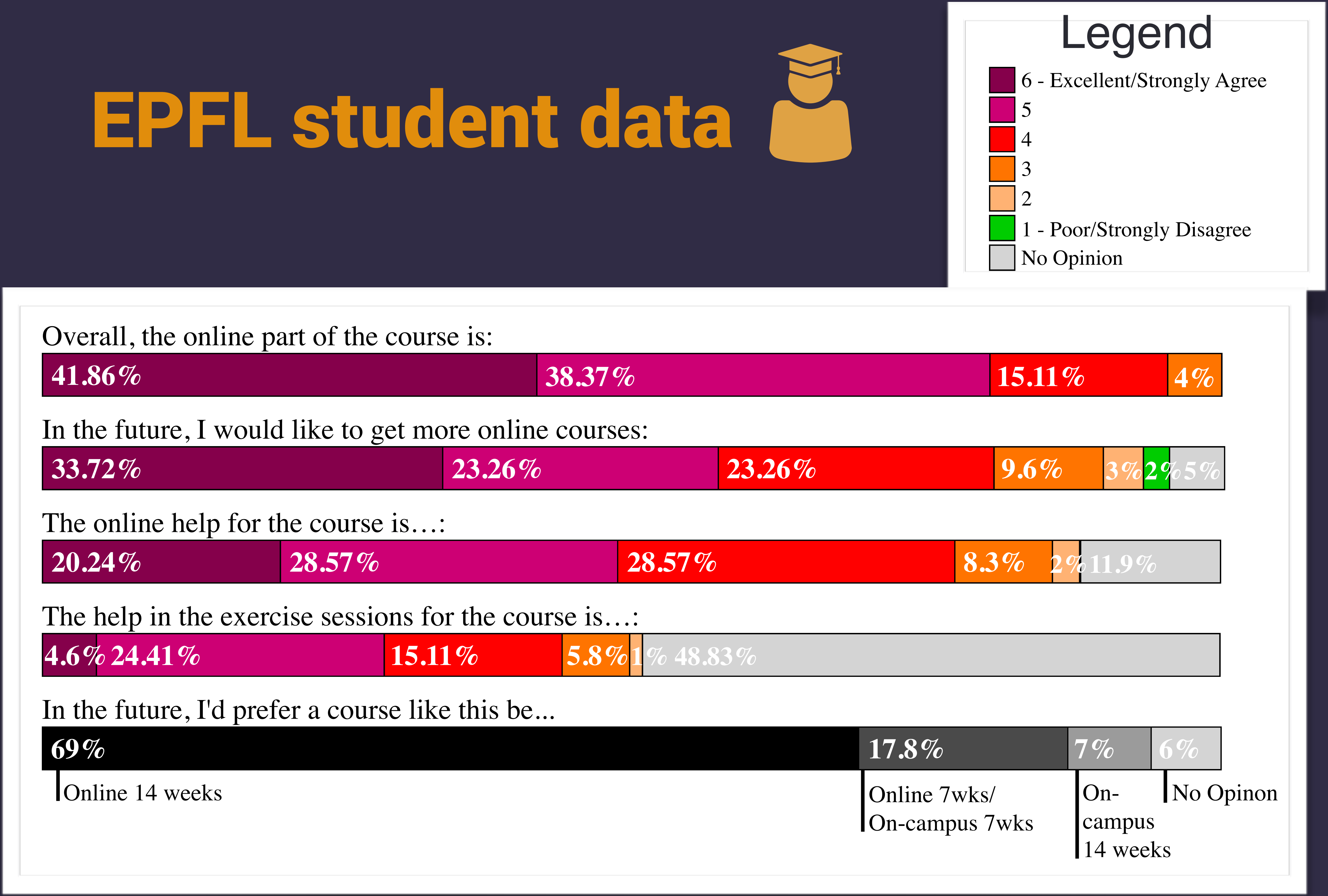
dat	updating with fall 2013 statistics	3 years ago
html/resources	Pushing to github	5 years ago
project	Pushing to github	5 years ago
src/main/scala/progfun	updating with fall 2013 statistics	3 years ago
.gitignore	Pushing to github	5 years ago
README.md	README update	3 years ago

EPFL student data

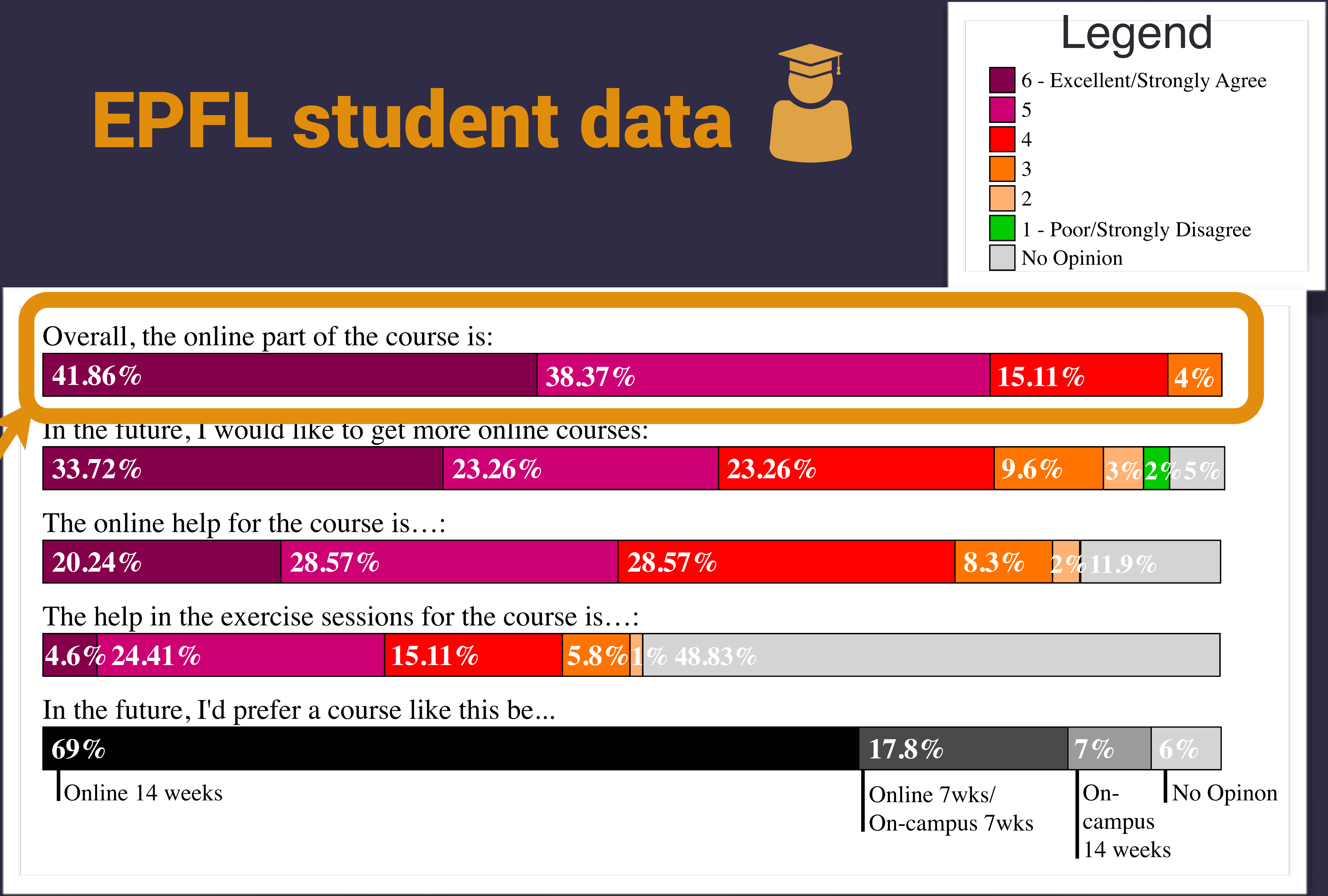
Post-course survey:

Given to EPFL students who took both the MOOC as well as the regular on-campus in-person course.

EPFL student data

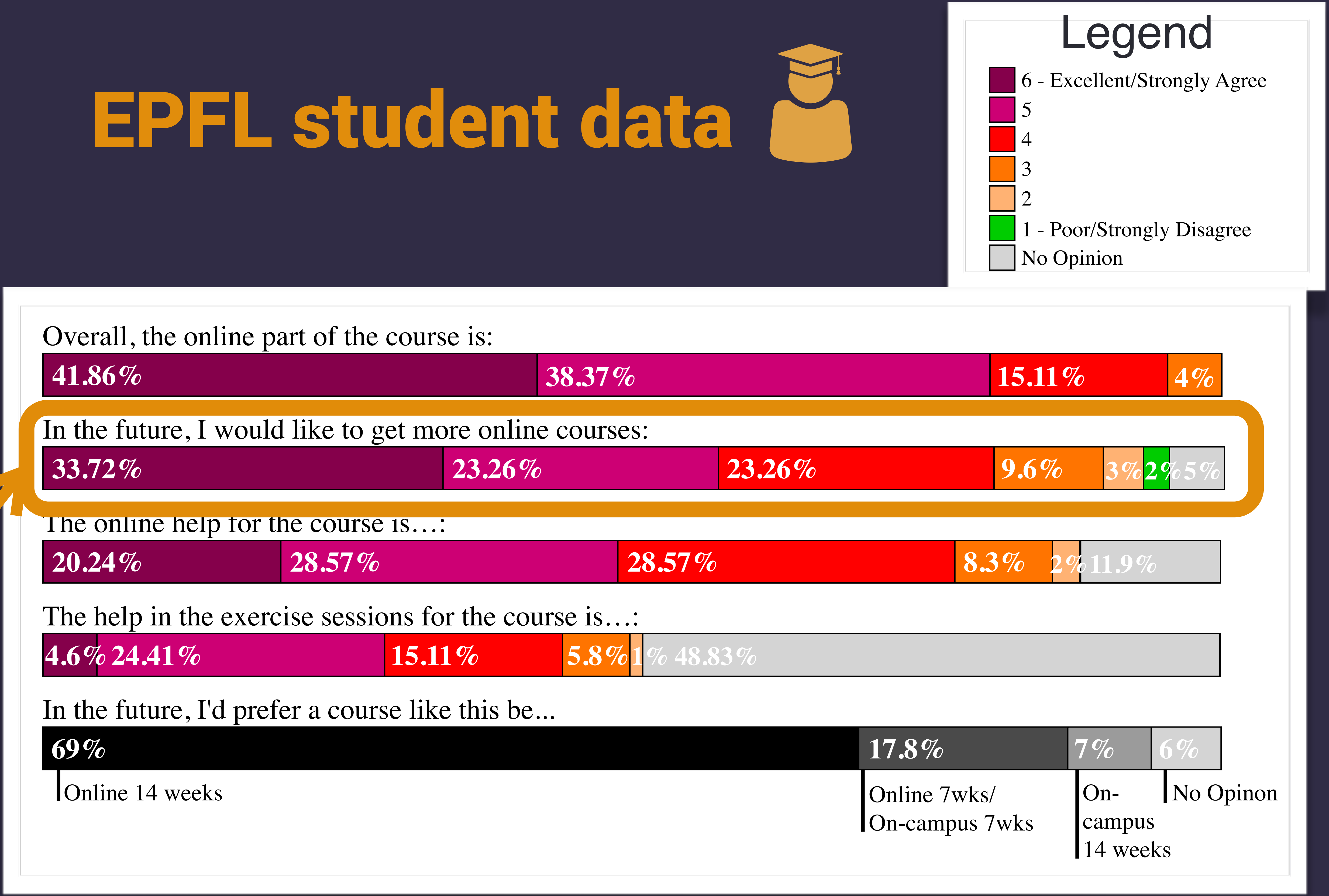


EPFL student data



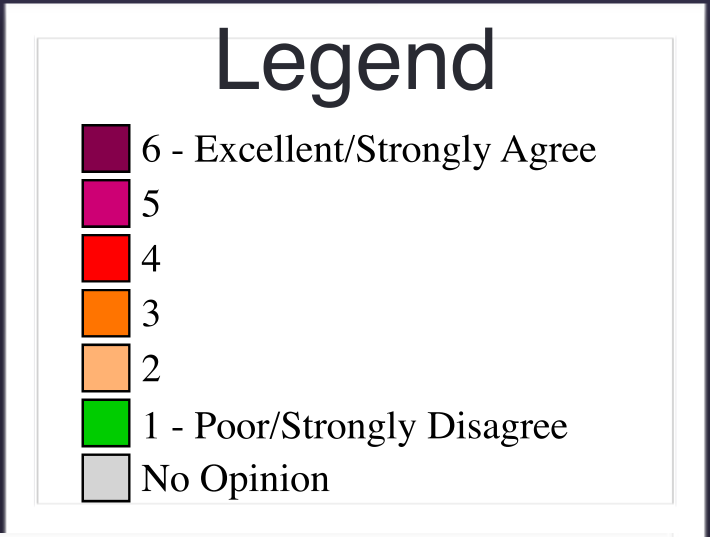
~80% of students think the course was very good or excellent

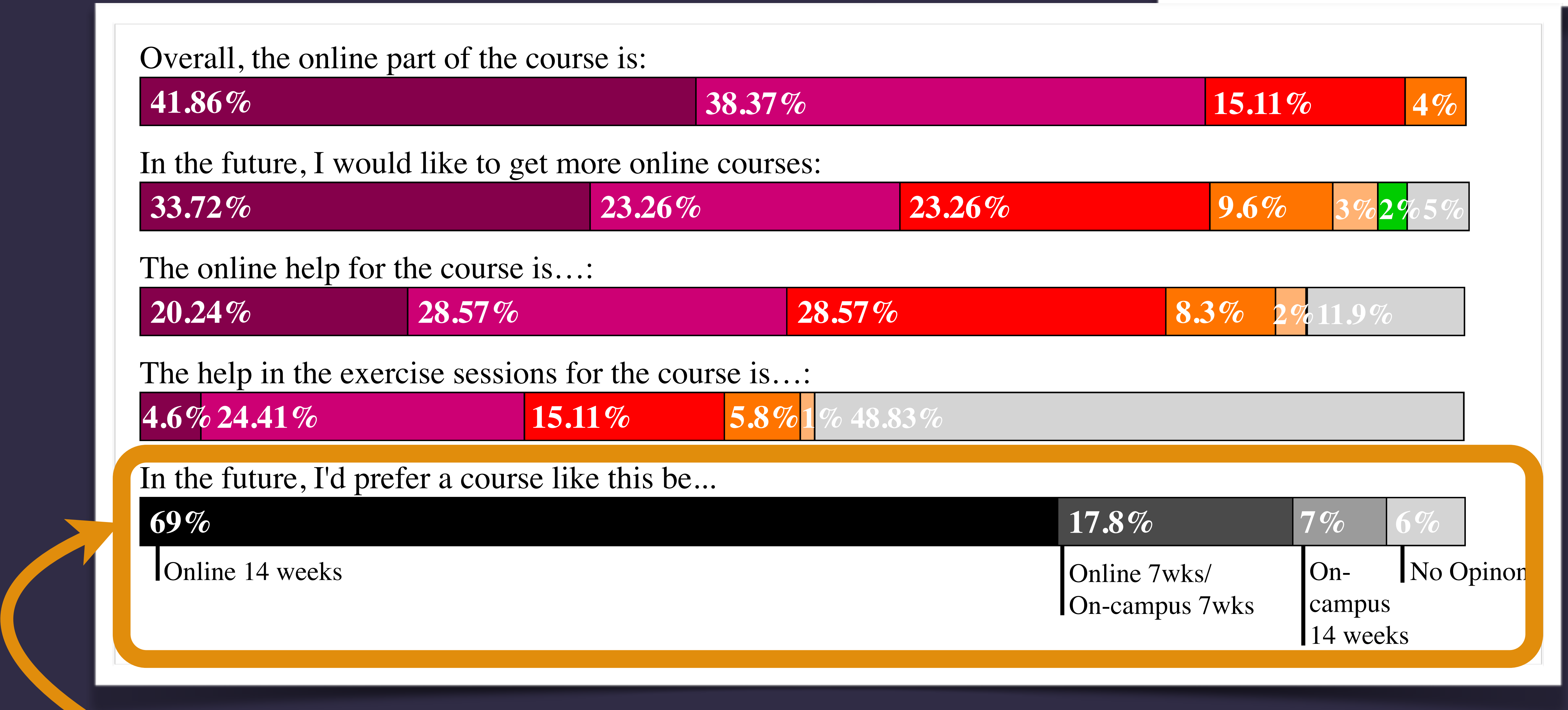
EPFL student data



~58% of students would like more online courses in the future

EPFL student data





~69% of students would like their entire course to be online, with no on-campus component

SO,

students seemed to overwhelmingly prefer the MOOC version of the course. In fact, students even preferred the MOOC forums to the exercise sessions.

Test performance remained the same, course ratings remained high.

The result?

✓ **Happier EPFL Students**
Good performance,
high course ratings

✓ **Uptaken and depended-on by professionals in industry**

Conclusions

between 2012-2013

✓ **Positive experience for all**

Both professionals and students alike had positive learning experiences.

✓ **Highest rate of retention for a course our size**

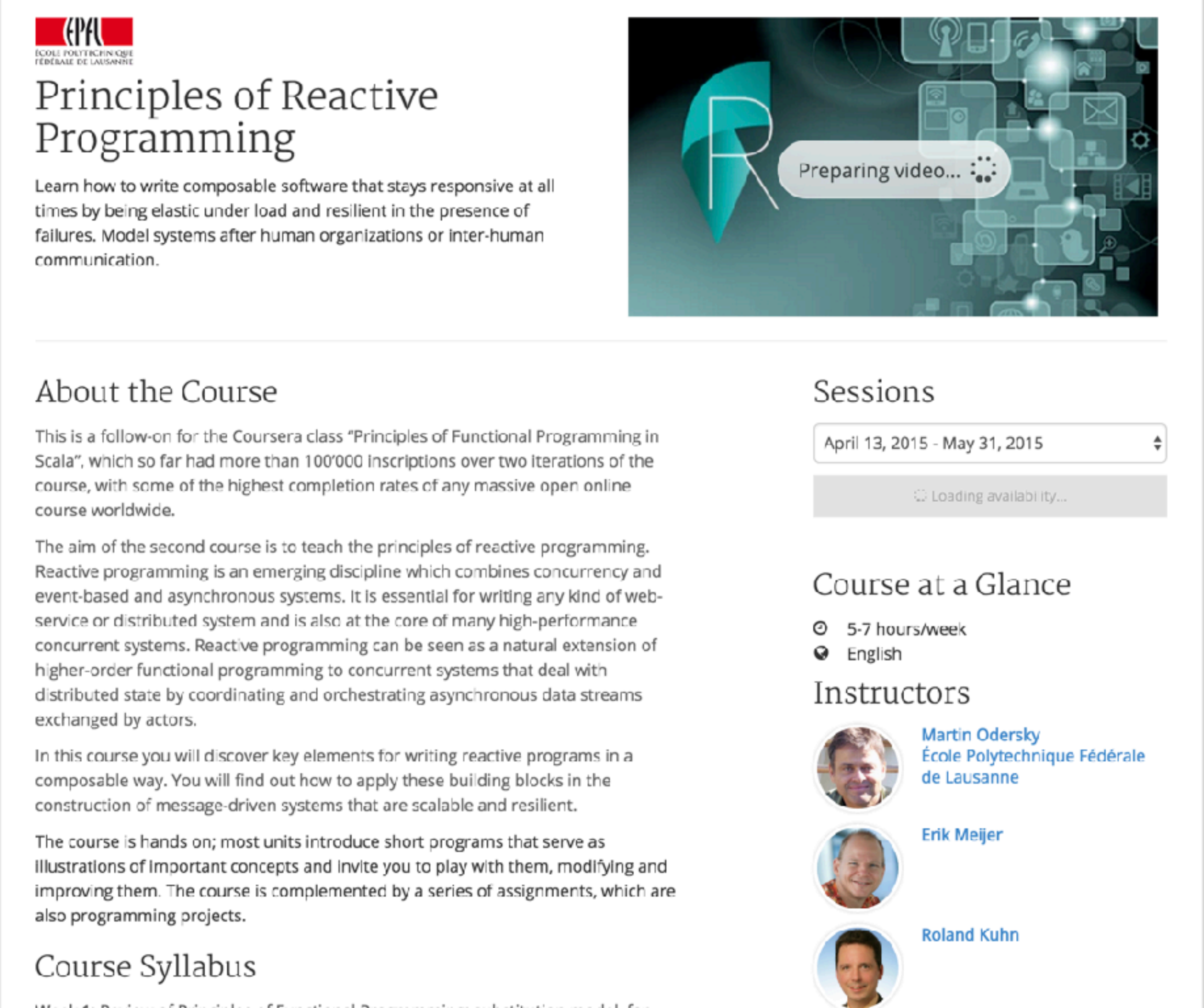
Our foray into MOOCs continued...

2013:

New MOOC,

Principles of Reactive Programming

~67,000 registrants in first run



Principles of Reactive Programming

Learn how to write composable software that stays responsive at all times by being elastic under load and resilient in the presence of failures. Model systems after human organizations or inter-human communication.

About the Course

This is a follow-on for the Coursera class "Principles of Functional Programming in Scala", which so far had more than 100'000 inscriptions over two iterations of the course, with some of the highest completion rates of any massive open online course worldwide.

The aim of the second course is to teach the principles of reactive programming. Reactive programming is an emerging discipline which combines concurrency and event-based and asynchronous systems. It is essential for writing any kind of web-service or distributed system and is also at the core of many high-performance concurrent systems. Reactive programming can be seen as a natural extension of higher-order functional programming to concurrent systems that deal with distributed state by coordinating and orchestrating asynchronous data streams exchanged by actors.

In this course you will discover key elements for writing reactive programs in a composable way. You will find out how to apply these building blocks in the construction of message-driven systems that are scalable and resilient.

The course is hands on; most units introduce short programs that serve as illustrations of important concepts and invite you to play with them, modifying and improving them. The course is complemented by a series of assignments, which are also programming projects.

Course Syllabus

Week 1: Review of Principles of Functional Programming: substitution model, for-

Sessions

April 13, 2015 - May 31, 2015

Course at a Glance

- 5-7 hours/week
- English

Instructors

Martin Odersky
École Polytechnique Fédérale de Lausanne

Erik Meijer

Roland Kuhn

Our foray into MOOCs continued...

2013:

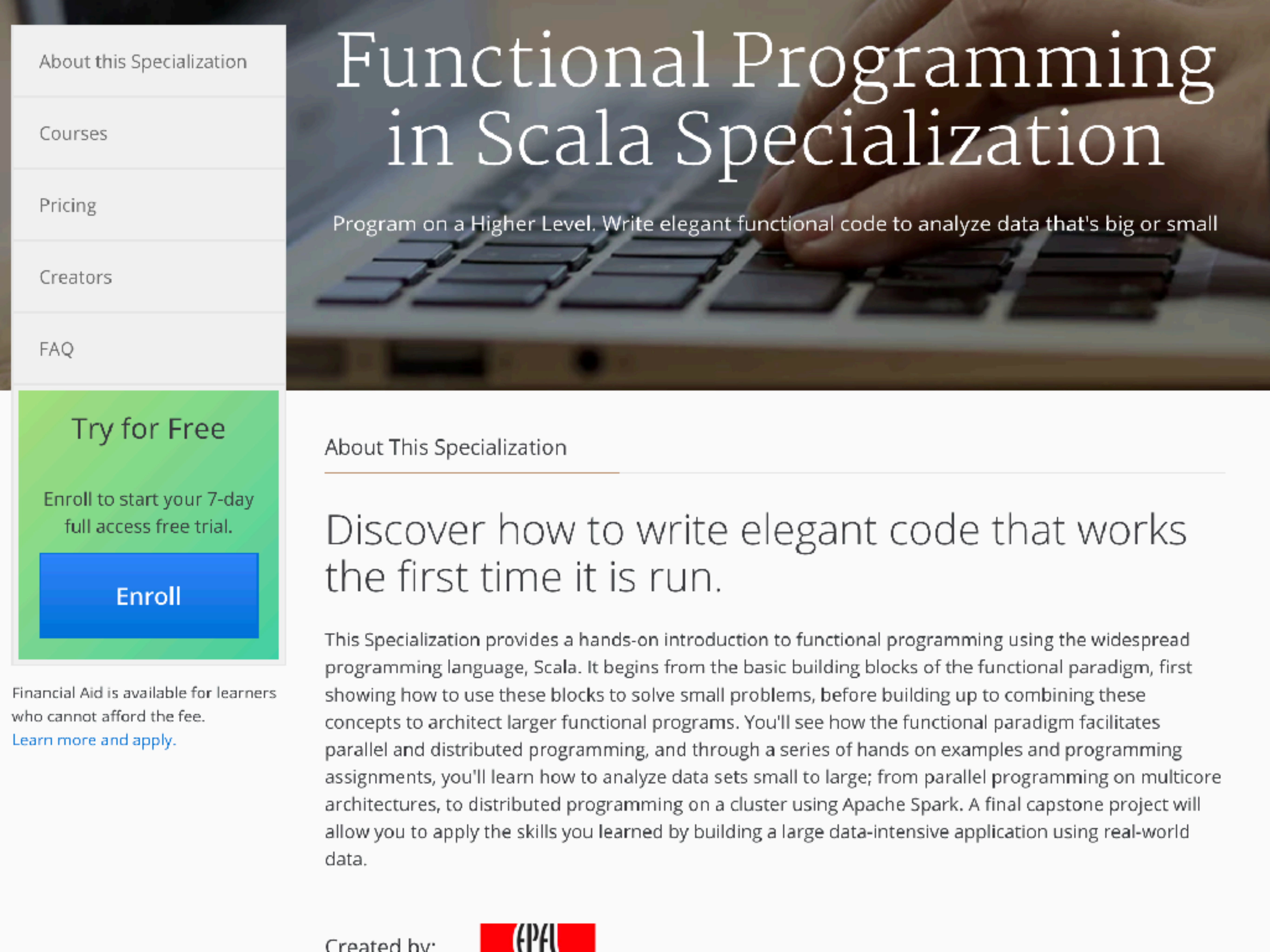
New MOOC,
Principles of Reactive Programming
~67,000 registrants in first run

2016/2017:

2 new MOOCs + capstone project bundled into a Scala mini-degree on Coursera

Parallel Programming
Big Data Analysis with Scala & Spark

400,000 registrants in first year
for courses in mini-degree



Functional Programming in Scala Specialization

Program on a Higher Level. Write elegant functional code to analyze data that's big or small

Try for Free

Enroll to start your 7-day full access free trial.

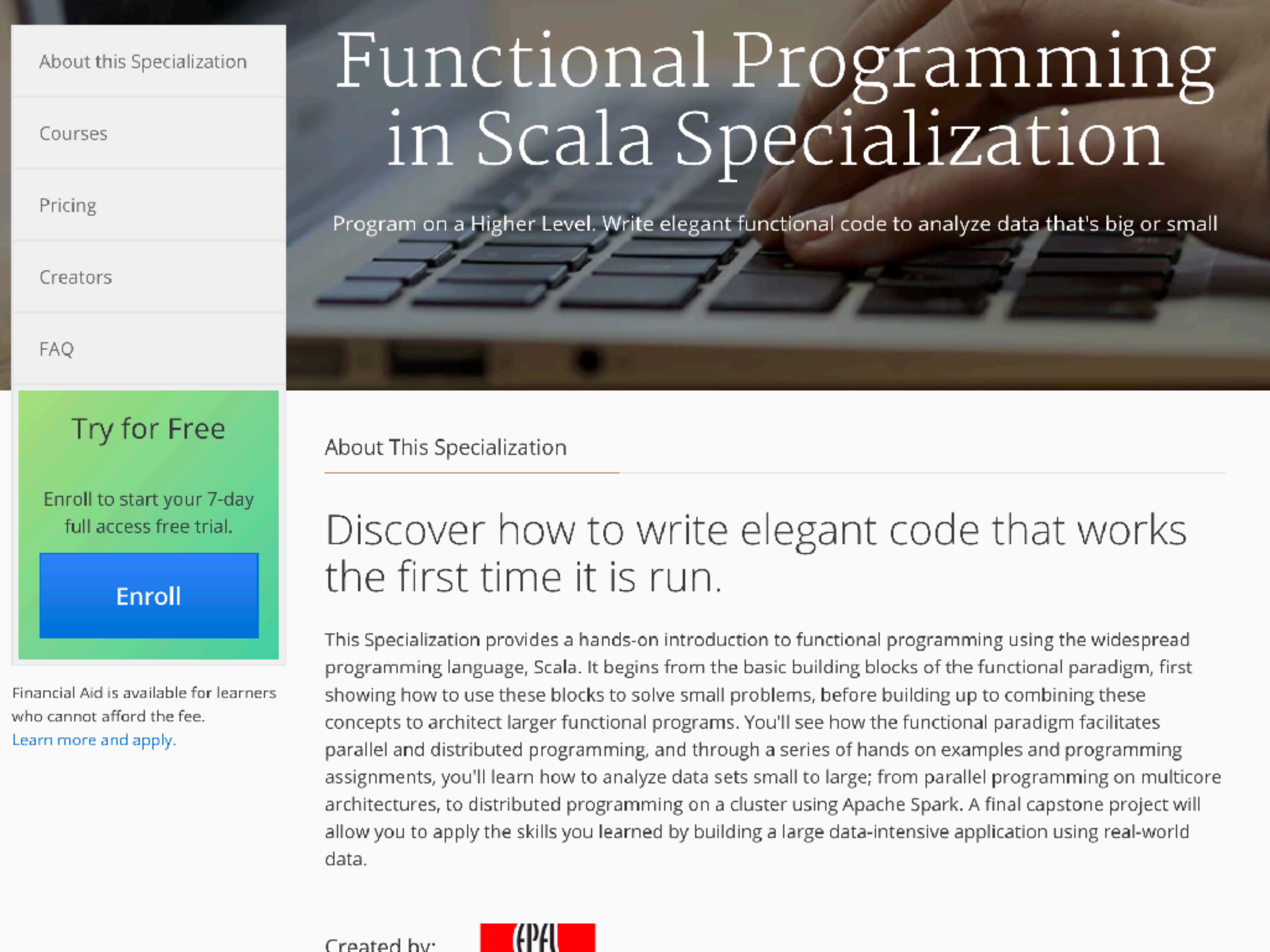
Financial Aid is available for learners who cannot afford the fee.
Learn more and apply.

About This Specialization

Discover how to write elegant code that works the first time it is run.

This Specialization provides a hands-on introduction to functional programming using the widespread programming language, Scala. It begins from the basic building blocks of the functional paradigm, first showing how to use these blocks to solve small problems, before building up to combining these concepts to architect larger functional programs. You'll see how the functional paradigm facilitates parallel and distributed programming, and through a series of hands on examples and programming assignments, you'll learn how to analyze data sets small to large; from parallel programming on multicore architectures, to distributed programming on a cluster using Apache Spark. A final capstone project will allow you to apply the skills you learned by building a large data-intensive application using real-world data.

Created by:



Functional Programming in Scala Specialization

Program on a Higher Level. Write elegant functional code to analyze data that's big or small

Try for Free

Enroll to start your 7-day full access free trial.

Financial Aid is available for learners who cannot afford the fee. Learn more and apply.

About This Specialization

Discover how to write elegant code that works the first time it is run.

This Specialization provides a hands-on introduction to functional programming using the widespread programming language, Scala. It begins from the basic building blocks of the functional paradigm, first showing how to use these blocks to solve small problems, before building up to combining these concepts to architect larger functional programs. You'll see how the functional paradigm facilitates parallel and distributed programming, and through a series of hands on examples and programming assignments, you'll learn how to analyze data sets small to large; from parallel programming on multicore architectures, to distributed programming on a cluster using Apache Spark. A final capstone project will allow you to apply the skills you learned by building a large data-intensive application using real-world data.

Created by:

Financials?

Financials?

Well, it works*

*mini-degrees, that is.

Financials?

Well, it works*

*mini-degrees, that is.

~2 million USD brought in in first fiscal year.

- Granted, how the money is actually split is a whole different issue.

Take aways…

Take aways…

Autograding + a tight feedback loop is key to retention. ★★★★★

Recent results [1] at LAK’17 arrive at the same conclusion.

[1] Follow the Successful Crowd: Raising MOOC Completion Rates through Social Comparison at Scale, LAK’17

Take aways...

Autograding + a tight feedback loop is key to retention. ★★★★★

Recent results [1] at LAK’17 arrive at the same conclusion.

Would I do it again in 2017?

[1] Follow the Successful Crowd: Raising MOOC Completion Rates through Social Comparison at Scale, LAK’17

Take aways...

Autograding + a tight feedback loop is key to retention. ★★★★★

Recent results [1] at LAK'17 arrive at the same conclusion.

Would I do it again in 2017?

Not sure.

Did you notice I didn't have data to show after 2013?

[1] Follow the Successful Crowd: Raising MOOC Completion Rates through Social Comparison at Scale, LAK'17

Take aways...

Autograding + a tight feedback loop is key to retention. ★★★★★

Recent results [1] at LAK'17 arrive at the same conclusion.

Would I do it again in 2017?

Not sure.

Did you notice I didn't have data to show after 2013?

The MOOC-provider landscape has drastically changed

[1] Follow the Successful Crowd: Raising MOOC Completion Rates through Social Comparison at Scale, LAK'17

Thank you!

Questions?