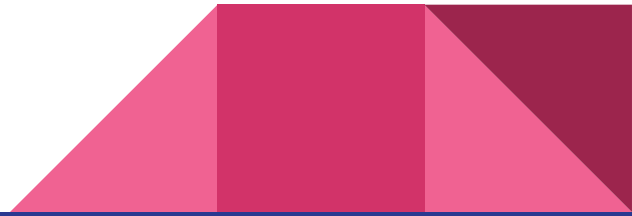
Learning Erlang Socially Over the Internet

TFPIE - 22 June 2017

Stephen Adams
University of Kent
sa597@kent.ac.uk

What is Erlang?

- Functional
- Message-passing concurrency
- Actor model
  - Lightweight processes and message passing
- Massively scalable and fault tolerant
- “Let it fail” error handling
- Developed in 1989 by Joe Armstrong, Robert Virding, and Mike Williams at Ericsson

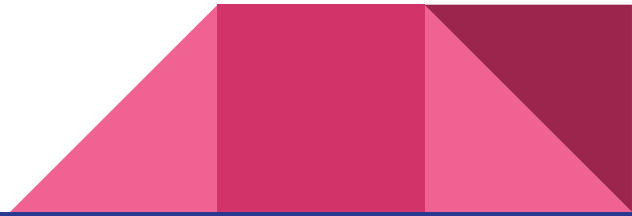


What is FutureLearn?

- MOOC platform owned by the Open University, UK.
- Launched in 2013
- 127 worldwide partners
  - Universities and other institutions (museums, libraries, etc.)
- Over 6 million users

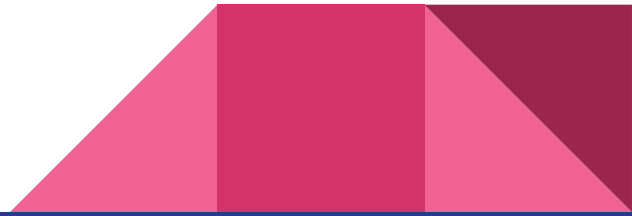
Overview

- Erlang at Kent
- MOOCs at Kent
- The Curricula
  - Functional Programming in Erlang
  - Concurrent Programming in Erlang
- Building MOOCs in FutureLearn
- The Participants
- What Happened
  - Common behaviours
- Future Work & Conclusion



Erlang at Kent

- Long history of research in functional programming
  - Wrangler - Erlang refactoring tool
  - Release - Building tools to make Erlang massively scalable
  - Prowess - Property-Based Testing for web services
- Stage 2 module taught in Erlang - CO545
  - “Functional and Concurrent Programming”
  - Principally taught in Erlang
  - Haskell covered towards the end
  - Erlang chosen for:
    - Both concurrent and functional
    - Clear concurrency model
    - Small set of key constructs



MOOCs at Kent - The Pilot MOOC

- Our first MOOC was a “Beacon Project” in 2015
  - One of a number of projects for the University’s 50th anniversary
- Created a three-week pilot MOOC
  - Material adapted from the first part of CO545
- Taught using the University’s Moodle site
- Short two to twenty minute video lectures
- Assignments for students
- 500 people signed up: capped at that number
- Recruitment mainly through twitter.

MOOCs at Kent - Lessons Learned

- A specialist MOOC platform would be preferable over Moodle
  - Although there were no complaints about the Moodle platform …
  - … and indeed some compliments about lecture playback.
- Time commitment was high
  - #1 reason people didn’t finish the course
  - Many people were planning to complete the course after it had finished
- The majority of people were interested in a full 6 week course
  - Functional and concurrent programming in Erlang.

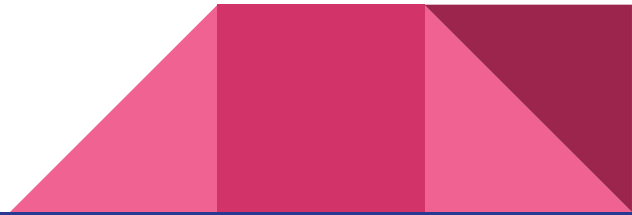
MOOCs at Kent - Developing for FutureLearn

- Positive experience from the pilot MOOC
- Wanted to include concurrency
- Decided to teach two three-week courses rather than one six week course
- Intended for someone familiar with programming
  - Though not necessarily a functional language
- First course would prepare someone with no Erlang experience for the second course

The Curricula

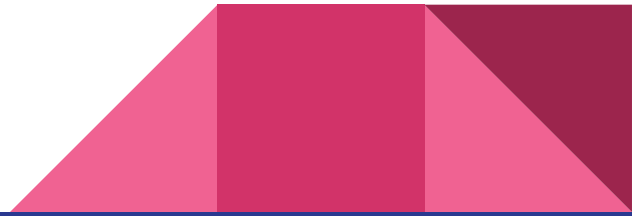
Curriculum - Functional Programming in Erlang

1. Getting started with Erlang
    a. Basic syntax (variables, pattern matching, functions)
    b. Data (atoms, tuples, lists)
    c. Recursion and tail recursion
2. Lists in Erlang
    a. Pattern matching lists
    b. Defining functions over lists
3. Advanced functional programming
    a. Higher-order functions
    b. Lambdas
    c. Functions as data



Curriculum - Concurrent Programming in Erlang

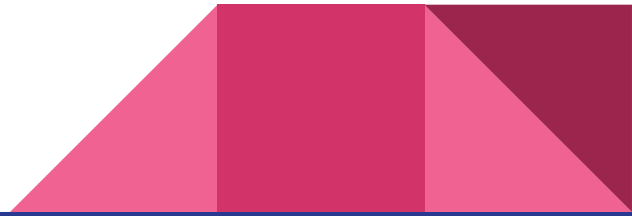
1. Concurrency - nuts and bolts
    a. Processes and messages
    b. Mailboxes
2. Concurrency - making code robust
    a. Process lifecycle
    b. Process linking
    c. Supervisors
3. Scaling it up
    a. Distributed Erlang
    b. OTP



Building a FutureLearn MOOC

Building MOOCs in FutureLearn

- Each week is broken into activities
  - Activities help suggest the learners pace
  - Organise steps
- Steps are the smallest section of the course
  - Each step support comments, and replies to comments.
- Teaching steps
  - Convey information to the learners
  - Interacting in discussion is optional
- Doing steps
  - Actively engage learners
  - Require interaction to pass



Learning steps - Articles

2.6

YOU'VE COMPLETED 0 STEPS IN WEEK 2

Upgrade

Defining functions over lists in practice

0 comments

Now we'll deploy the template from the previous videos to make recursive definitions. The aim of these exercises is to familiarise you with defining functions over lists in Erlang, in particular the different way that functions can combine the elements of a list.

Combining list elements: the product of a list

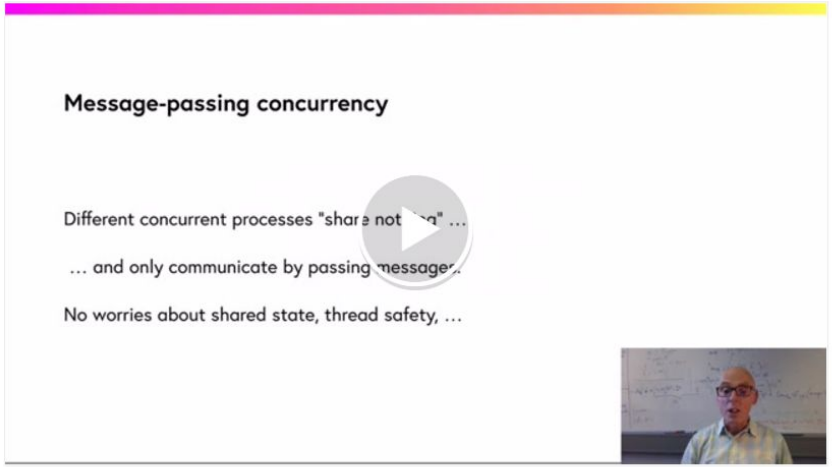
Using the template from the last session, define an Erlang function to give the product of a list of numbers. The product of an empty list is usually taken to be 1: why?

Learning steps - Video/Audio

2.2

YOU'VE COMPLETED 1 STEP IN WEEK 2

Upgrade



View transcript

Download video: standard or HD

Building robust systems

14 comments

Doing steps - Discussion

1.18

YOU'VE COMPLETED 1 STEP IN WEEK 1

Upgrade

Erlang concurrency so far

We've come to the end of our case study of the frequency server.

We'll conclude this week with another two presentations from Joe Armstrong, who will talk about how to build various concurrency primitives using Erlang and provide a canonical example of client-server.

But first, now is a good time to pause and reflect. How are you getting on with Erlang concurrency? You might like to discuss how it compares to your experience of concurrency in other languages. Or you could comment on any particular features that you find difficult or confusing, or those that you find intuitive or elegant. Remember to 'like' those comments that you find especially helpful, or that raise important questions.

Doing steps - Assignments

2.20

YOU'VE COMPLETED 0 STEPS IN WEEK 2

Programming challenge: indexing a file

This activity gives you a chance to try out a larger problem using what you have already learned about functional programming in Erlang. After you have completed it, you will be able to get feedback on it from other learners, and to give feedback on others' work too.

In solving this problem you'll need to think about the different stages of processing of the data: you begin with a list of lines, each of which will need to be broken into words, and those lines (and words) will need to be associated with the corresponding line numbers. So, thinking about useful intermediate stages – and helper functions – should help you to make progress in solving the problem.

Doing steps - Quizzes & Tests

Question 3

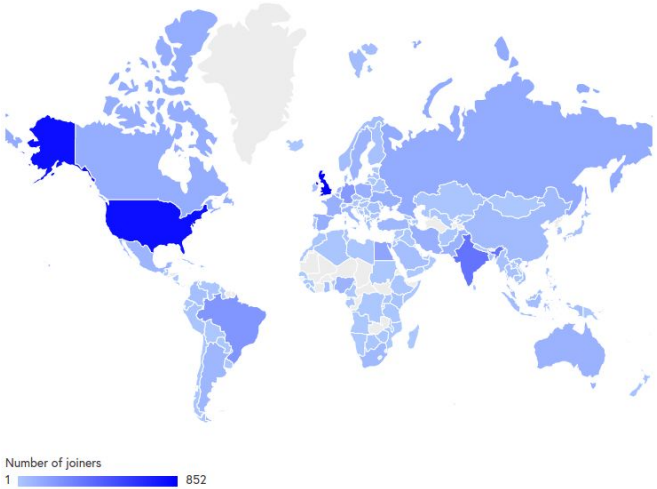
Suppose that you have to remove all punctuation and change capital letters to lower case. How could this be implemented?

- ◯ map followed by foldr
- ◯ map
- ◯ map followed by filter
- ◯ filter

Who Participated

Who Participated

- 5,642 people enrolled in Functional Programming in Erlang
- 1,965 people enrolled in Concurrent Programming in Erlang
- Majority from the US and UK but 149 countries were represented



Prior Experience

C/C++	52
Java	51
JavaScript	49
Python	41
Ruby	31
Haskell	26
PHP	23
C#	14
Erlang	11
Scala	11

Most popular Languages

Haskell	26
Erlang	11
Scala	11
Elm	9
Clojure	8
F#	4
Scheme/Racket	4
ML/OCaML	2
Idris	1
Emacs Lisp	1

Functional Language Experience

Interest in Functional Programming

[redacted] Follow 15 MAR

Hi,

I am [redacted] I doing this course to understand more about functional programming and the use of Erlang in building reliable distributed systems.

Like Reply Bookmark

[redacted] Follow 10 MAR

Hi, I'm from Australia. Saw Erlang was available on the Raspberry Pi. Had never heard of it before, but I have been interested in functional programming, so , I thought , why not? LOL.

Like Reply Bookmark

[redacted] Follow 02 MAR

Hello all, dev w/ 5+ years industry experience. Ruby is my language of choice but more and more I'm finding myself attracted to functional languages/concepts (e.g. Elixir, React). Excited to learn and get started.

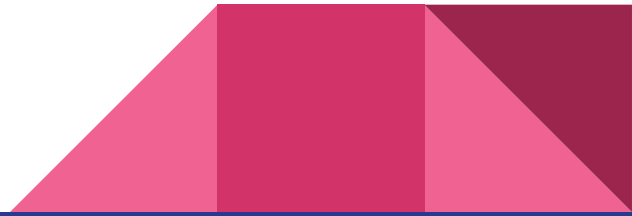
Like Reply Bookmark

What Happened

What Happened

- Wanted to facilitate social learning
- Two of us “vs” thousands of learners
- Needed learners to engage each other for help

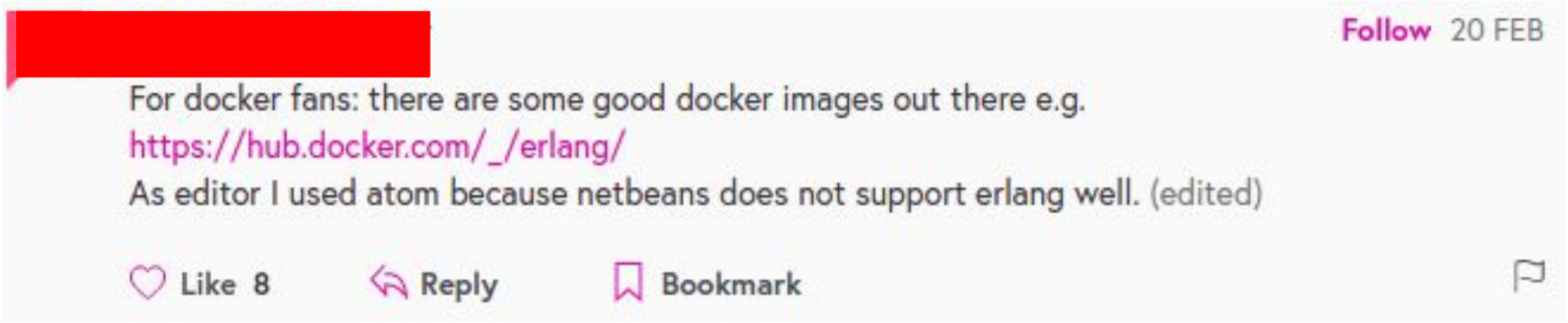
- We provided no automated feedback
- Recorded short feedback video at the end of each week



My Role

- Monitor discussions
- Contribute where necessary
  - People shouldn’t get “left behind”
  - Large discussion with a common misunderstanding
  - Highlight underappreciated comments
- Avoid jumping in too soon
  - Don’t want people to “expect” us to intervene
  - Don’t want to “settle” the issue and kill good discussions
- About one hour per day

People like to share (1)



People like to share (2)

[redacted] Follow 24 FEB

If you are running Ubuntu 16.10, it is as easy as :
sudo apt-get install erlang-base

Other versions of Ubuntu should be similar, as will Debian based distributions.
This installs Erlang/OTP 18 [erts-7.3.1.2], although the latest version appears to be 19, all the examples so far work.

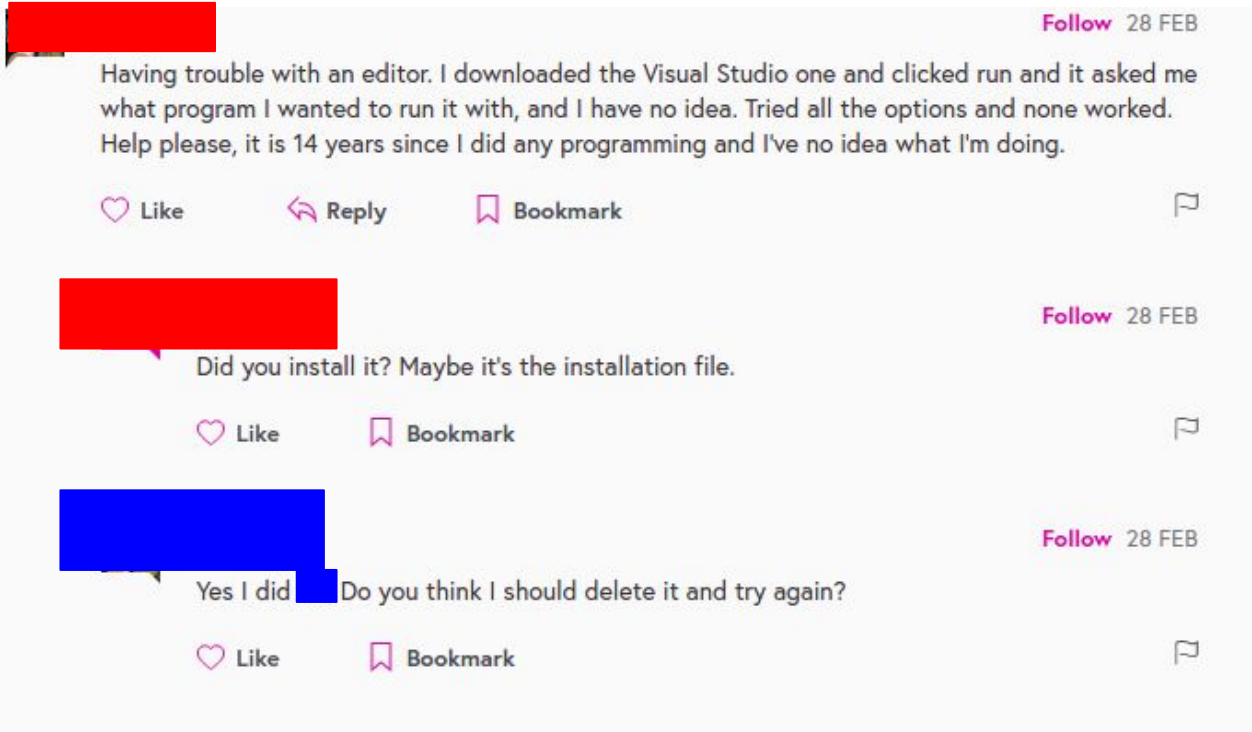
To use the latest version, you must first uninstall the above packages if you prevoiusly installed them (sudo apt-get uninstall erlang-base):
Get the latest package:
wget https://packages.erlang-solutions.com/erlang/esl-erlangAVOUR_1_general/esl-erlang_19.2.3-1~ubuntu~yakkety_amd64.deb

Install it:
sudo dpkg -i esl-erlang_19.2.3-1~ubuntu~yakkety_amd64.deb

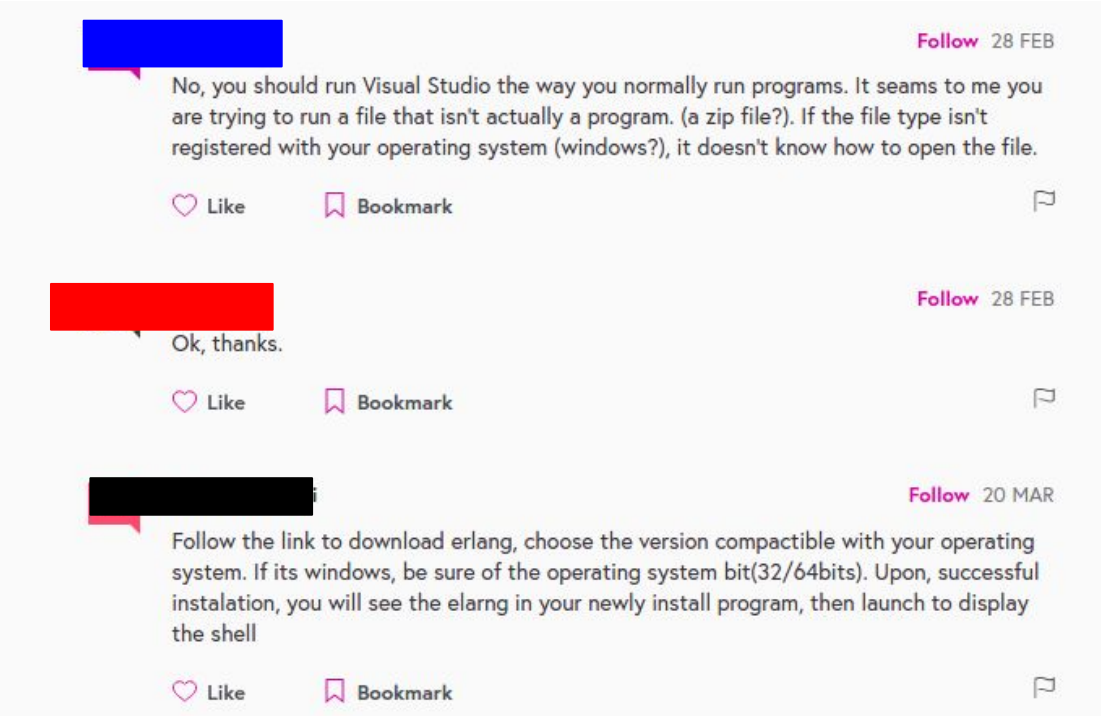
This will install Erlang/OTP 19 [erts-8.2.2] (edited)

Like 1 Reply Bookmark

People like to help (1)

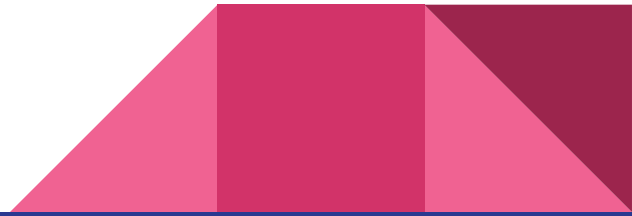


People like to help (2)

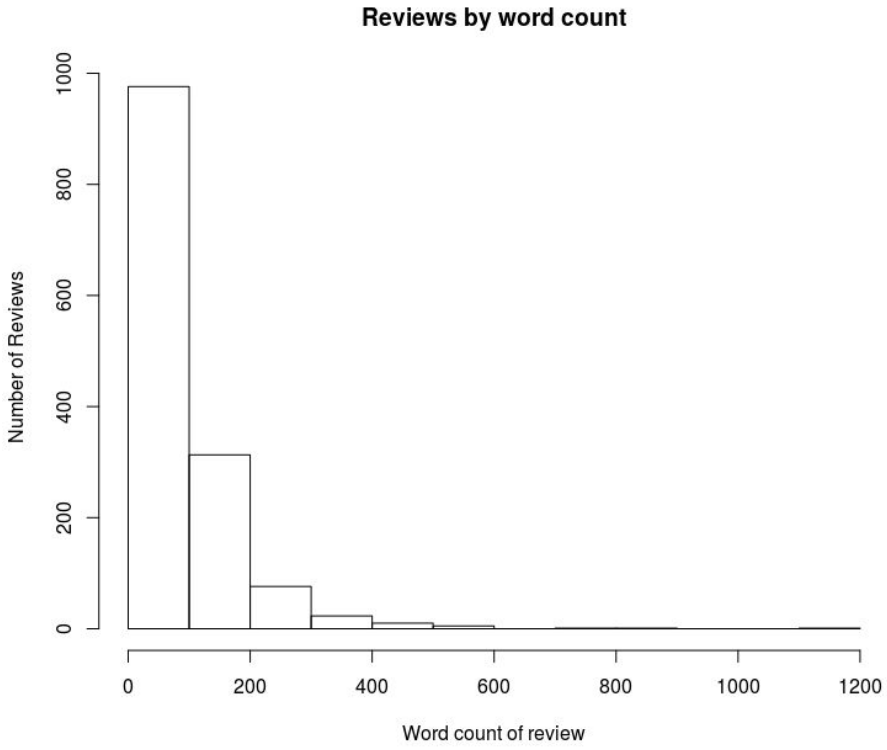


Lack of Feedback

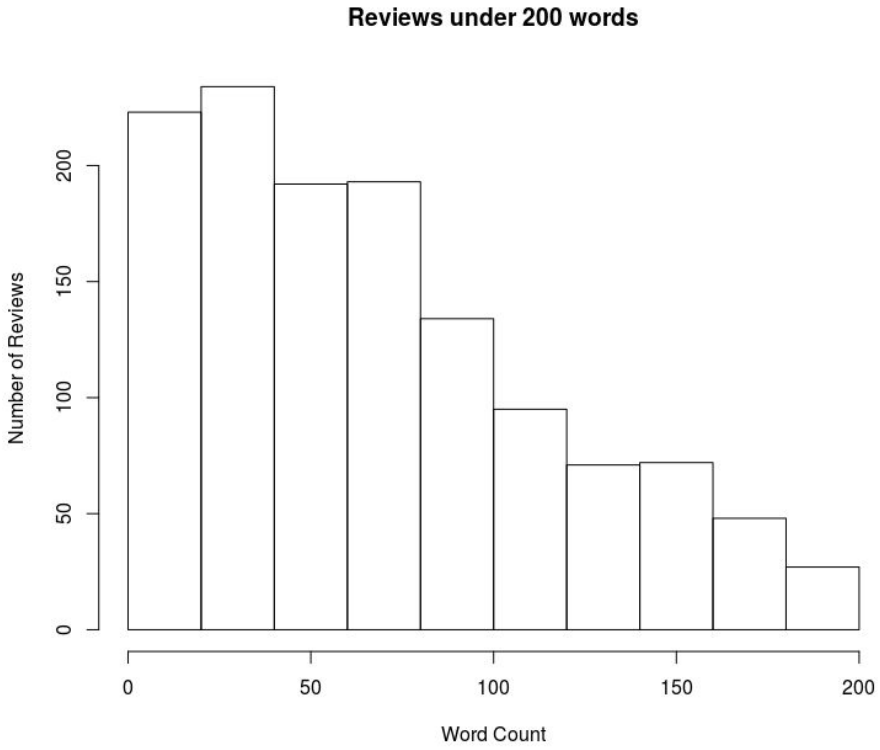
- Feedback on informal exercises was sporadic
  - Best results came from specifically asking for feedback from peers
- Most assignments only had one reviewer
- Quality of reviews was mixed



Reviews by word count

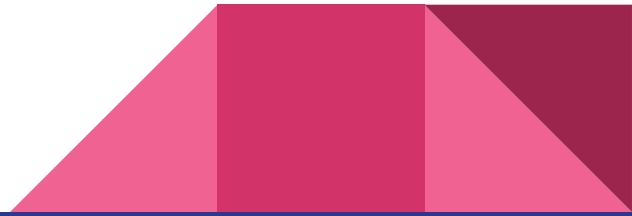


Reviews by word count



Reviews

- 1412 reviews in total
- 117 reviews greater than 200 words long
- Longest review was 1170 words long
- Most assignments only had a single reviewer



Reviews - Example

- “Your file name does not match name of module. You have provided only one version of bits function.”
- “OK”
- “I think I could add `when` case to `area({triangle, {A, B, C}}) ->` function instead of adding if. You have created nested tuple. It is nice but I think that inlined tupled is more readable for me: {triangle, A, B, C}”

Reviews - Gist (1)

review.md — Raw

1. It looks like you aren't handling some of the punctuation correctly. When I try your solution out on my test text, I see some punctuation still on the end of some of the words in the index (see the example below).
2. It also looks like you are skipping some words.

When I try your solution on a file with a single line:

```
Hi, there!  What's up?  See you later.
```

I get the following index:

```
[{"later",[{1,1}]},{"there!",[{1,1}]},{"what",[{1,1}]}]
```

There are only three words, and one of them has an `!` on the end. That is a consequence of trying to enumerate all the "bad" characters. Simon did that in his solution in the video, but a much better solution in my opinion is to use guards to select a subset of the "good" characters in different function clauses, e.g. A-Z, and construct words character by character. Here's an example:

```
normalize(L) ->
    normalize(L, []).

normalize([], Acc) ->
    reverse(Acc);
normalize([X|Xs], Acc) when X >= $A, X =< $Z ->  %Caps, so
    normalize(Xs, [X+32|Acc]);  %convert to lower case
normalize([X|Xs], Acc) when X >= $a, X =< $z; X=:=32 -> %Lower case and spaces, don't
    normalize(Xs, [X|Acc]);   %make any changes.
normalize([_|Xs], Acc) ->  %Anything else...
    normalize(Xs, Acc).  %skip 'em.
```

You are already doing part of that work in your `no_cap()` function--you might as well go all the way.

If I add some more text with blank lines to see if your line numbers stay in sync with the lines:

Reviews - Gist (2)

```
Hi, there!  What's up?  See you later.

Preceded by a blank line, so this is line 3.
```

I get the index:

```
[{"blank",[{3,3}]},
 {"later",[{1,1}]},
 {"line",[{3,3},{3,3}]},
 {"preceded",[{3,3}]},
 {"there!",[{1,1}]},
 {"this",[{3,3}]},
 {"what",[{1,1}]}]
```

Nice job handling the blank lines. A suggestion: I can see that "line" has two entries for line 3. The word does appear twice on the same line, but I think one reference per line is nicer. What do you think? To get a single {3,3} run for "line", you could call `nub()` on the line numbers to remove duplicates before converting the line numbers to runs.

Hey, I just realized I never got around to dealing with a word like `"what's"` in my solution. I see you converted it to `"what"`. Nice. I just tried it, and another option is to loosen the guard in `normalize()` above to let in `$'` as a good character--then I got `what's` in my index instead of `whats`, which is terrible. :(

Ah, I just saw this:

```
convert_to_tuple_with_line_number(remove_small_words(nocaps_all(split_words(Line))), LineNumber, []).
```

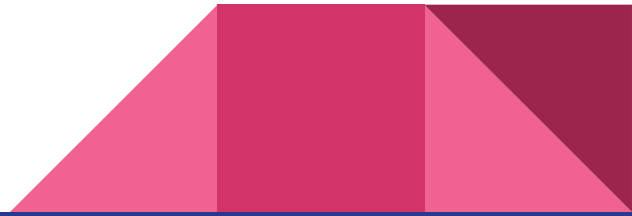
That would explain the missing words!

I like your function composition--it reads rather well. I've been shying away from adopting that style in my code, but I think I'll make an effort to try that style in some exercises.

See you in class!

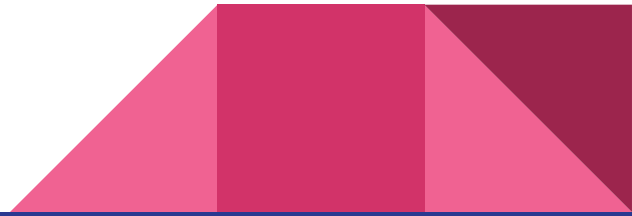
Reviews

- No complaints about the quality of the feedback

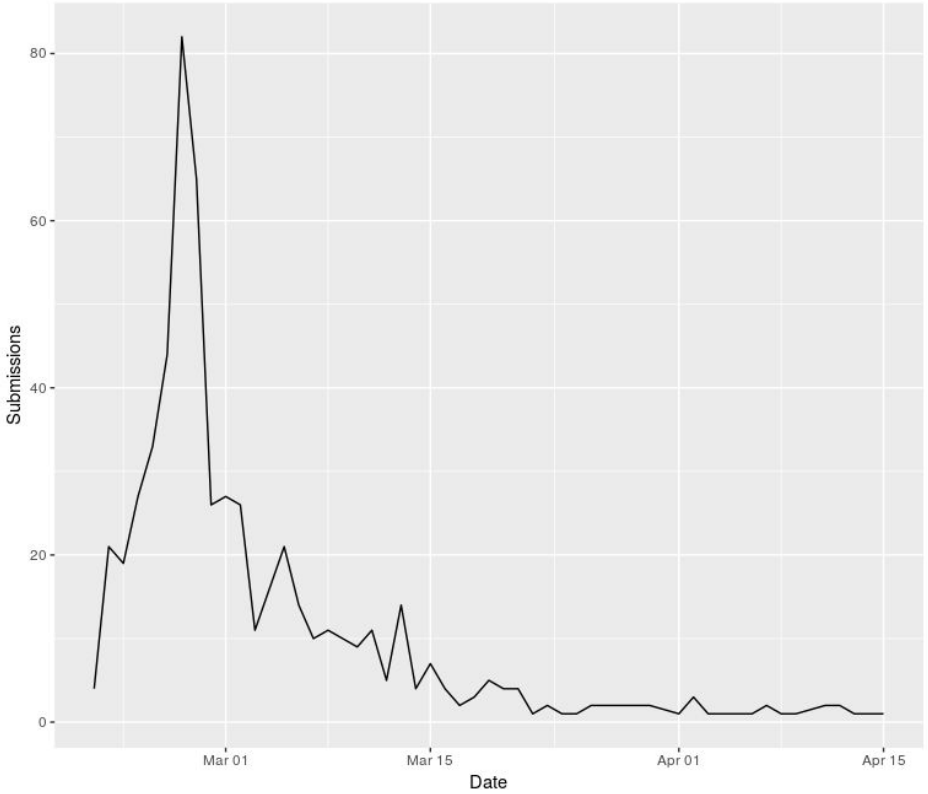


Pacing

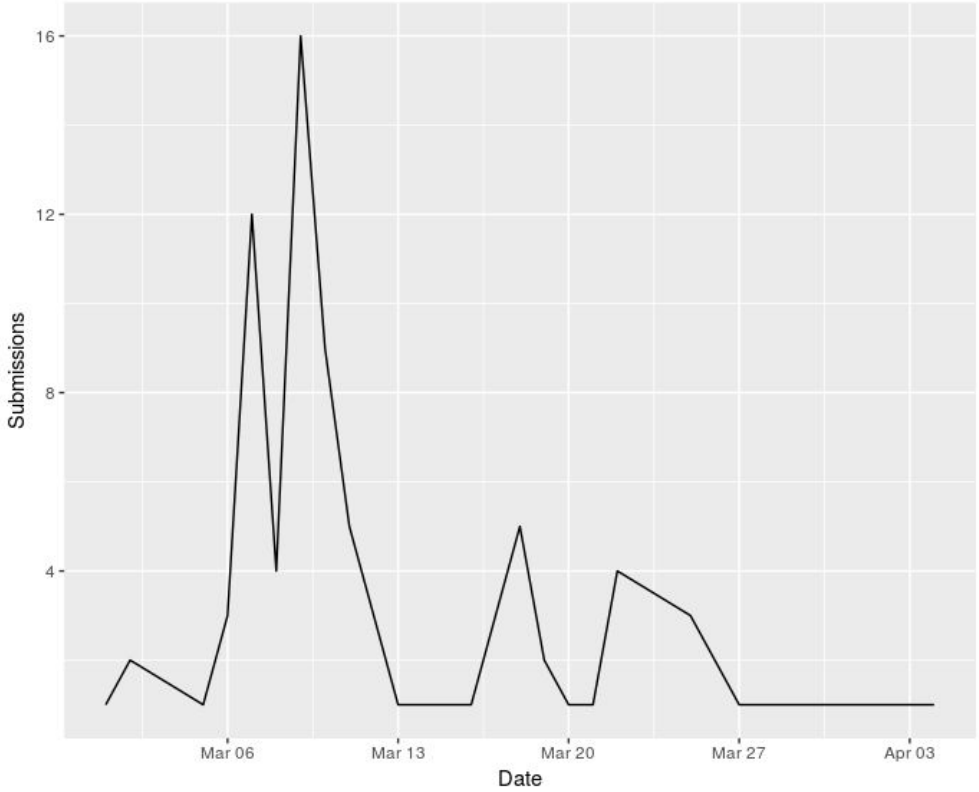
- 8 & 15 days fastest completion times
- “Core group” of 4-10 people finish each week’s work very quickly
- They normally stuck around and commented on other’s work
- The rest on time or behind course schedule



Submission Timeline - Assignment



Submission Timeline - Exercise



Submission Timeline - Exercise

[redacted] Follow 18 MAR

When we did tail recursion we 'hand rolled' it so that we understood the principle - rather than just saying 'use foldr and don't worry your little heads about it'. I've tried to do the same here and just get absolutely nowhere. I could copy any number of foldl solutions posted here and sort of understand them (i.e. not at all) so I tried to hand roll it - and I'm just totally baffled by the 'syntax' of this mess. Everything I try just comes back with something totally meaningless - like you say, let it fail, so yeah, the course has failed.

I'm outta here - and I can't say it's been much fun.

(Oh, and thanks for the really helpful source code files which are all on one line and if you split them up for Windows the compiler error line numbers are meaningless.)

Like Reply Bookmark

[redacted] Following 18 MAR

I remember reading about an interview with a CalTech Astro Physicist who pursued computer programming as a career. He was asked if he ever regretted expending so much effort getting his PhD in Astro Physics. He replied, "No. Sometimes when I'm trying to solve a programming problem, I can get discouraged and I feel frustrated, and then self doubt creeps in, and I start feeling like I'm not smart enough to figure out a solution, and I feel dumb. Then I remember, 'Hey, I've got a PhD from CalTech in Astro Physics. I'm not dumb.' Some things are just hard."

7 (edited)

Like Bookmark

[redacted] Following 18 MAR

By the way, foldl() is trivial to define yourself.

7 (edited)

Like Bookmark

Submission Timeline - Exercise

[redacted] Follow 19 MAR

But my point is I can't see how to do it here. Sure, some things are hard but gloating about how easy YOU find them is not terribly productive.

Like Bookmark

[redacted] Following 20 MAR

@[redacted],

I'm sorry, I think I might have misunderstood your issue. I thought you were trying to define iterate() without using any version of foldl().

Without lists:foldl(), I used four hand rolled functions to define iterate()--one of which was my compose() function. I found defining a single iterate() function with nested function calls to be impossible, and, like you, I found that the error messages weren't helpful.

However, after the fact I tried to figure out a simpler solution for iterate(), and I discovered that defining my own foldl() function was extremely easy. Did you try that? In my opinion, defining my own foldl() was easier than any exercise/assignment we've had in class. It's essentially just map(), which simply applies a function to every element in a list (where in our case the list will contain a function, like F = fun(X) -> X+2 end, repeated N times, e.g. [F, F, F, F]).

With my own foldl() function in hand, I was able to come up with a much simpler solution for iterate(). If you already tried to define your own foldl() to enable you to solve iterate(), then my advice is moot. (edited)

Like Bookmark

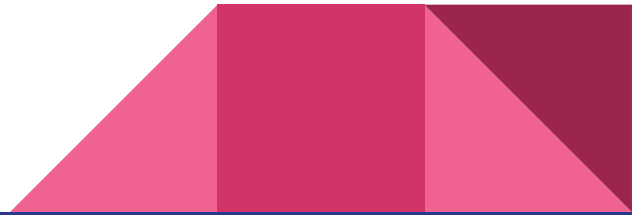
[redacted] Follow 21 MAR

Thanks [redacted] Actually my problem was trying to get anywhere with the 'compose a list of functions' problem which is probably 'trivial' - but I often find that things that are supposed to be obvious aren't. Maybe I just over think them.

Like Bookmark

Future Work

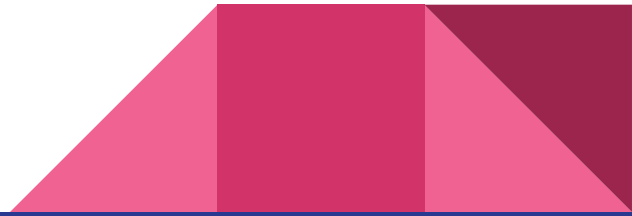
- Teaching Functional Programming in Erlang again
  - Interest was high enough for a second offering
  - Began on 19th of June
- Development of a third course
  - Focused on the Open Telecoms Platform (OTP)
  - OTP is a set of libraries for developing distributed and fault tolerant systems
  - Working with industrial partners
- Make these three courses and a final assessment into a FutureLearn “Program”



Conclusion

- A MOOC based on social learning can work
- Overall people seemed happy with the course
- Survivorship bias makes it hard to judge
- Curious how behaviours change with smaller cohort
    - 5,000+ enrolled vs <1000

Thank you to Mark O’Connor



Questions?

What Happened - Participation

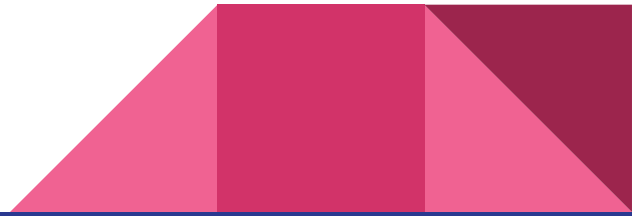
Category	**FP in Erlang**	**Concurrent Prog.**	*Notes*
Joiners	5,642	1,965	
Learner	3,858	1,117	68% & 57% of joiners
Active Learner	2,683	676	70% & 61% of learners
Social Learner	586	142	15% & 13% of learners
Fully Part. Learner	374	40	10% & 4% of learners

What Happened - Participation

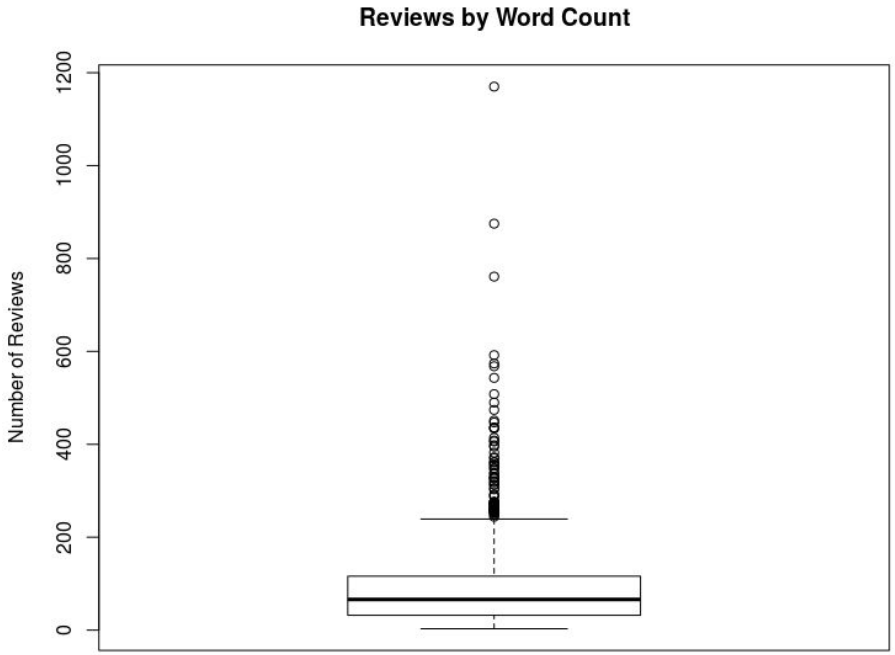
Category	FutureLearn Avg.	FP Avg.	Concurrency Avg.
Learner	50% of Joiners	68%	57%
Active Learner	81% of Learners	70%	61%
Social Learner	38% of Learners	15%	13%
Fully Part. Learner	21% of Learners	10%	4%

Other behaviours

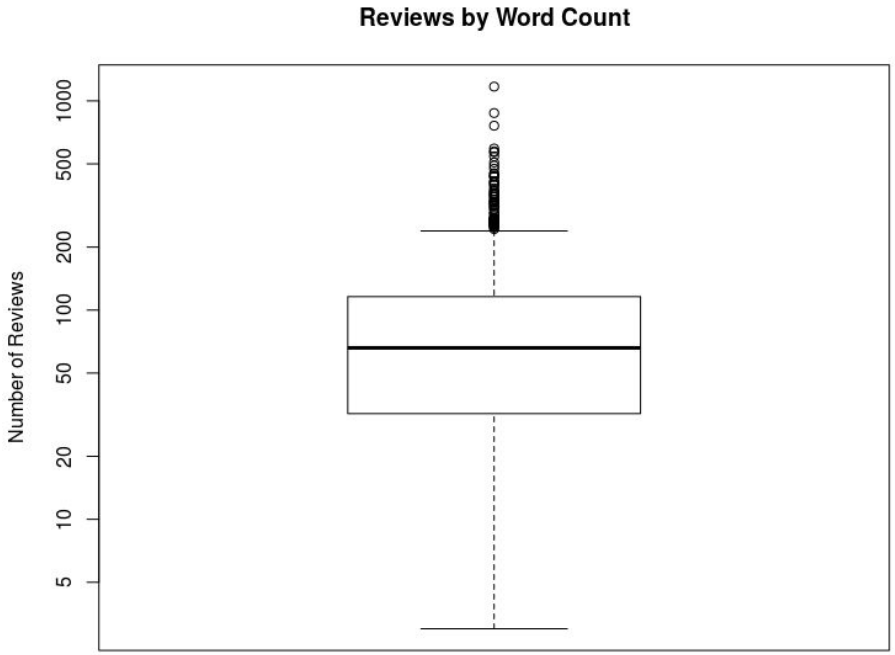
- Some people “policed” coding style and testing
- Mentors
- External resource sharing



Reviews by word count

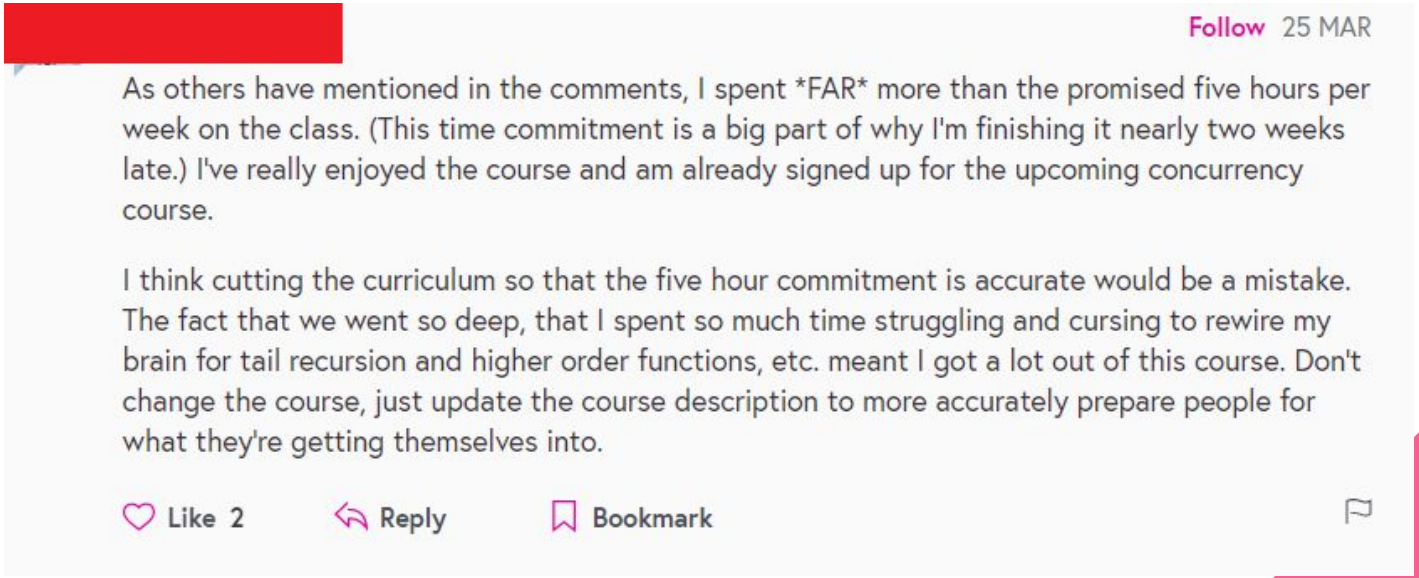


Reviews by word count



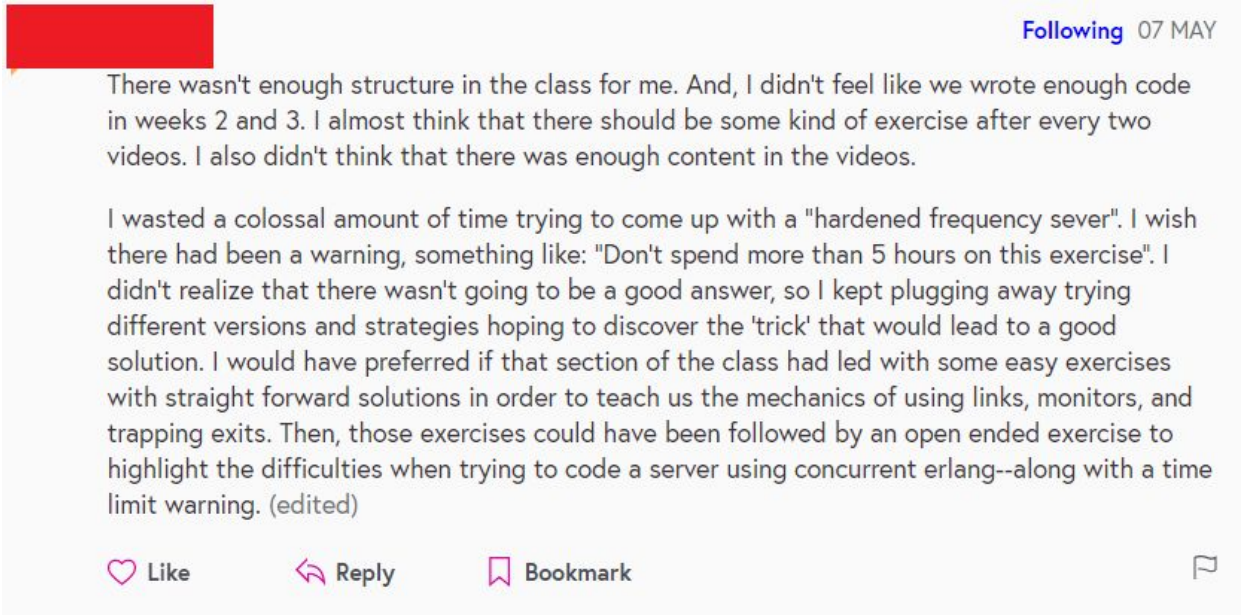
Feedback - Workload

- We estimated 5 hours a week
- Learners indicated that this was an underestimate



Feedback - Workload

- Concurrent programming in Erlang had more open ended exercises



What Happened

- Wanted to facilitate discussion
- At the end of each week we recorded a short feedback video
  - Clarified material that was causing difficulty
  - Gave input into discussions that had occurred
  - Handled issues that came up that week

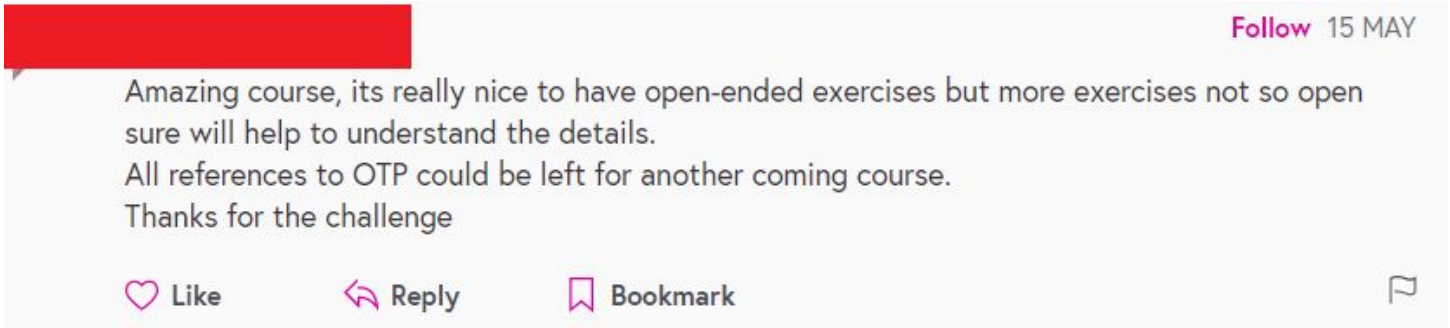
What Happened - Participation

FutureLearn categorises learners based on how they progress through and interact with the course.

- Joiners
- Learners - Visited at least one step
- Active Learner - Completed at least one step
- Social Learner - Left at least one comment
- Fully Participating learner - Completed all tests and 50% of the steps

Feedback workload

- People like open ended exercises in theory...



General comments

Thanks Stephen. Slides are fine as far as they go. I've made a few inessential changes, but I wonder whether we can give some more high-level feedback on the experience of the course, including

1. The experience of the assessments: what can we say about use of feedback from one person to another. How did people find the feedback? How did they find receiving the feedback?
2. A similar question about the more informal mechanism that we encouraged of commenting on submissions in the comments section?
3. The different roles played by people: I was thinking in particular about the people who took on an explicit mentoring role, but I am sure that there are other types of behaviour too.
4. Are there any general comments that we can make about the way that people interacted in the comments?
5. Anything that we can say about the pacing? We had people joining once the course was going … we also had people "lagging behind" …
6. Did we have any problems with getting people to install the tech for themselves? I wasn't aware of any, but we might just have had people fading out silently?

The general point I'm making, I guess, is whether we can give some high-level – albeit subjective – insights that go beyond what's in the simple data?

S.