Informatics Student Course Feedback 2017/18

http://www.inf.ed.ac.uk/teaching/surveys/2017-18

This report contains feedback from students about a course taught in the School of Informatics during the 2017/18 academic year, in response to the following questions:

- What would you say to students interested in taking this course?
- What did you find most valuable about the course?
- What improvements, if any, would you make to the course?
- Please add any other comments you have about workshops and tutors

Each course organiser receives this report as well as statistics on multiple-choice responses. All these reports, together with student feedback about individual members of teaching staff, are collected and sent to the Director of Learning and Teaching.

Please note that these are personal responses from individual students: some courses only have a few responses and a small sample can be unrepresentative.

Stereotyping and bias, especially unconscious bias, is a serious concern in any survey gathering personal responses. All students received the rubric below before completing the surveys, and you can read a brief introduction to issues of unconscious bias on the university web pages at http://edin.ac/2iypZBv

Rubric given to all students taking the end-of-course feedback survey

We value your opinions on the courses you take here at the University, as they allow us to shape future delivery and development. We welcome constructive comments about your courses, whether positive or negative, and ask you to give details about any issues in order to help the course organiser to understand and address them.

We encourage you to be aware of the potential for bias in the completion of these questionnaires, so we have developed resources which may be helpful to you:

- Equality, Diversity and Unconscious Bias (http://edin.ac/2iypZBv)

You also have a responsibility to provide feedback in a manner which does not breach the University's Dignity and Respect Policy:

- University of Edinburgh Dignity and Respect Policy (http://edin.ac/1Cq0VZY)

The results of the questionnaires will never be analysed in a way that seeks to identify individual students from their responses. However, should you wish to remain anonymous, please do not identify yourself in your answers to the survey questionnaire implicitly or explicitly.

Comments Report

What advice would you give to a student taking this course in future?

- Definitely spend more time on reading the online book than the yellow book. Which seems to me that is the book for supporting the beginners.
- Definitely try and complete the tutorials to get some practice and make full use of Piazza if you have any questions.
- Do all the tutorial questions before the tutorial - the tutorials are really useful if you actively participate. If you don't have much programming experience definitely go for a beginner tutorial
- Do all the tutorial work
- Do as much of the tutorial exercises as you can before the tutorial. It'll let you get a lot more help during the tutorial.
- Do more practice of programming.
- Do the tutorial exercises, learn to use google to help find solutions.
- Do the tutorial work, and read the slides at a bare minimum.
- Do the tutorials early - approaching them as challenges that you want to complete is so much better for learning and enjoying it than getting code down as fast as you can.
- Do the tutorials in advance.
- Do the workshop exercises thoroughly.
- Do to read the book if you are a totally beginner.
- Don't take it as an outside course unless you are planning to take computing further i.e. get a degree in it. It takes focus off of your core subjects due to the sheer workload and it only being worth 10 credits.
- Don't think you know functional programming because you know Java or other. Also don't brag in the first lectures asking questions about advanced topics. They don't apply and you'll confuse beginners. Ask the lecturer in person at the end instead or at the advanced tutorials. The syntax will make no sense at first but just listen and read Lipovaca, and everything will be revealed to you.
- Enjoy and appreciate how awesome the programs we write in tutorial questions are! eg encrypting messages, screenscraping, FSMs Enjoy!
- Go to all tutorials, especially at the beginning of the course. Make sure you have the fundamentals down and understand them or you won't get much more of the later parts of the course.
- Haskell is fantastic!!!!
- Have a basis understanding on programming
- Google is GOOOOOOOOOOD!!
- If you don't have time, prioritise doing tutorial exercises before reading. Otherwise you will waste one hour every week in tutorials.
- If you don't know what you are doing move to beginner tutorials, pay attention to the work and do it. Trying to do tutorials without knowing what you are doing is not gonna help you with anything
- If you have no basis in functional programming - it's going to be challenging. If you do, then go to the advanced tutorials, because nothing will be particularly new otherwise.
- It’s quite time consuming.

Comments Report

What advice would you give to a student taking this course in future? (continued)

- Make sure to leave plenty of time to do the tutorial work before your tutorial, as sometimes though some of the problems may seem confusing or insurmountable, working over them over a few days and collaborating with peers can be a very helpful learning aid and allow you to get the most out of the course. Also pay attention to Don Sannella in lectures and you might just learn something.
- Persevere with tutorial activities even if you have to skip questions and look at solutions to make sure you fully understand how they work
- Practice and do the tutorial exercises
- READ THE TEXTBOOK CHAPTERS BEFORE EACH HOMEWORK, and do the optional material in the tutorials.
- Read the book if you get stuck on an assignment. Learn You A Haskell has all that you need and more.
- Read the book.
- Show them where to find a terminal on your computer.
- Stay up to date with tutorials
- Stay ahead of the tutorial exercises by about a week.
- Study slightly ahead of the course would makes the course much more easier
- To enjoy it.
- Work through all the questions from the tutorial sheets including the optional material because it will increase their programming skills a bit further than the average.
- Find a smart student to help you if you’re not of robot intelligence.
- just enjoy it

Comments Report

What did you find most valuable about the course?

- Alternate way of approaching programming.
- Being able to learn a new language in a paradigm that I've never programmed before was fascinating.
- Doing the optional exercises helped to improve my programming skills even outside of Haskell and overall functional programming taught me a few things I did not know before (e.g. how Haskell is not a useless language). I also liked how we briefly looked to complexity analysis.
- Don's teaching!
- Enthusiastic and energetic lecturer made it really fun to be in lectures and learn the course content. The tutorial questions were very helpful for putting the theory we'd learnt into practice.
- Hmm. There's a fair bit - the tutorials are awesome, especially when they take on an almost project form. The lectures, too, are very informative and inspiring.
- How it introduces new programming language to students
- I definitely learned a lot from tutorials.
- I improved my programming skills and learned how to solve a problem more efficiently (in most cases).
- I learnt Haskell
- Interesting to be able to learn a completely new topic
- It helped me learn functional programming, in Haskell. Which I couldn't do before at all.
- Learning about programming
- Learning new ways to think in functional programming
- Lectures. Tutorials are also well structured.
- Let me get in touching with programing.
- Some aspects were glanced over and not great detail taken in them
- The ability to be able to write functional programming
- The after class revision.
- The course itself.
- The fact that we start programming in a functional language, even without having any previous experience on imperative ones.
- The lecture notes were fairly well done and the Tutorials were useful as they gave a small scale setting that really helped firm up any difficult material.
- The lecturer was clear and also entertaining.
- The lectures and lecturer were really good and the tutorial exercises had challenging questions too.
- The optional exercises
- The optional exercises from tutorials were quite challenging and allowed me to develop my problem solving skills.
- The skill to simply the program.
- The tutorial exercises were the best part: they really allowed you to develop your programming skills.
- The tutorial work was the most valuable thing for me as I found it provided necessary practise and application of the concepts and ideas presented in the lectures.
- The tutorials
- The tutorials and tutorial questions really developed my problem solving skills which will be useful for my engineering course
- Tutorial activities directly related to lectures with helpful feedback in tutorials Tutorial exercises, tutorials, advanced tutorials
- Tutorial exercises
- Tutorials (2 Counts)

Comments Report

What did you find most valuable about the course? (continued)

- Tutorials going over work
- Woosh
- Thinking about programming in a totally different way
- not sure

Comments Report

What improvements, if any, would you make to the course?

- Get a stable editor/environment
- Have more practical demonstrations in the lecture, I find it hard to learn programming by just reading lecture slides or by just listening to the lecturer.
- Having more tutors available in labs, especially over the 1st few weeks
- I didn't enjoy the method of teaching where the entirety of my knowledge depended on my reading of the subject. As although I did read the material thoroughly, I was still unsure about some complex topics and would have preferred it if these were taught in detail during the lectures. Tophat was an interesting way of gaining feedback from the student but I felt there was too much emphasis on tophat questions and would rather more emphasis on the understanding of difficult topics.
- I feel like the level of programming required to do well for this course is a bit too low. Even though the optional tutorial material is very challenging, the rest of the course doesn't require much work or preparation.
- I think a bit more active problem solving/active coding by the Professor could help a bit as sometimes I felt I had the tools but did not know the best way to approach a problem.
- I think it could be even more beginner friendly. There should be more help with initially downloading Haskell and Atom. Make clear what a terminal is. Also more help with getting used to the dice machines.
- I wish we did more complexity analysis, I know we will do so in second year but this is the sort of knowledge people who are looking for internships must have. If we did more to prepare us for employability as early as possible, we could deepen our understanding in the further years without missing out on opportunities to develop careers from the start of university.
- I'd suggest an open lecture idea in which students should interact more.
- It is good
- Make it deeper and harder
- Make it more challenging, cover more advanced topics (like get into monads more)
- May be give us more understandable practice in lesson that is linking to the problems we will do seeing in the tutorials. Be more friendly to the beginners.
- More challenging tutorial exercises.
- More focus on algebraic data types
- More help for beginners. More organised at beginning of course (atom was a nightmare)
- More help in labs
- More interaction in classes
- More optional exercises and maybe explain inputs and outputs more
- More optional stuff?
- More organised in regards to what software is needed to code on
- Perhaps make it a bit more general. For example, this course was focused too much in Haskell where it would be much better if it was focused generally how the techniques in Haskell are applied generally in functional programming and not only in Haskell. Furthermore, most people don't even know why Haskell will be useful or even, why they are learning it and they consider it as a waste of time. This, however, should be cleared up at the beginning of the course in greater details of why Haskell and generally functional programming is useful, what are their uses and, why should we learn functional programming and why is better and worse than imperative programming (which all of us are familiar).

Comments Report

What improvements, if any, would you make to the course? (continued)

- Speed of the course is sometimes too fast.
- Start of semester was confusing with Haskell setup and first week lab not well explained for beginners
- The lecturer should seem much more confident and organized and less hasty. Also, he clearly is aware of the different skills and needs of listeners, and he tries to cater for that, but often still fails to be effective because of the aforementioned problem.
- The tutorial exercises are sometimes too hard for beginners.
- Use something other than Atom, or have atom actually work on home all platforms. Caused a lot of hassle.
- Usually there is no room in the lab sessions and sometimes tutors are not present.
- change tutorials, i find them completely useless I have learned nothing from being forced to complete a series of exercises having had no clear instruction as to how to go about them. Unlike some people on this course I need to actually be taught something in order to do it. In my opinion tutorials should be changed to a 2 hour intimate lab session where someone goes through how to do the tutorial instead of being shoved an exercise sheet with no idea what to do. I'm worried I will fail this exam.
- https://www.haskell.org/hoogle/
- incredibly hard for complete beginners to integrate - in a beginners tutorial but most people in it have had programming experience and got 90%+ in the midterm test. Maybe more workshops specifically for people who are properly struggling
- none

Comments Report

Please add any other comments you have about workshops and tutors

- Having longer sessions of tutorials could help beginner friendly groups.
- I did not really use the labs and for the most part, tutorials served only to enforce deadlines for assignments. I think this is in no way anyone's fault, in fact I really like my tutor and tutorial group, just that programming is more of an individually learned skill.
- I like that there is optional material available.
- I was not in the beginner tutorial, which would have been more helpful for me, but I didn't bother to move.
- Lab demonstrator is not very approachable
- More extra interesting questions.
- My tutor was very helpful and friendly giving us good feedback and guidance.
- N/A
- Often only covered part of the work in tutorials
- Perhaps it would be useful for students to solve a few extra exercises cooperatively instead of just checking the solutions prepared at home. This way different approaches could be compared.
- Personally I'd prefer if we went slightly faster, but I can't see any way that could happen without missing out - so maybe (this is definitely not going to be possible, is it.) lengthen the time allotted to them?
- The advanced tutorials were much more useful than the standard tutorials. I found the tutorial exercises themselves to be very helpful. The tutorial time is limit, so that we cannot talk about many problems.
- The tutorials were incredibly helpful.
- The workshops were brilliant
- Tutorials are very helpful. At first in labs it is very unclear who the tutor is. For first students this can be intimidating
- Tutorials proved to be quite useless. Most of the time everyone could do the tutorial exercises, so we just read out the solution and agreed on it. If it hadn't been obligatory, I would have never attended. Tutorials didn't involve any active thinking because we just read out code we wrote days ago. It would have been useful, in my opinion, if the tutor had asked further questions related to exercise, to deepen our understanding on the subject.
- Tutorials were fantastic. Nikita was a great tutor and provided useful tips for the class test and Haskell in general. I like the format of them as you could ask questions very easily
- While the exercises were useful for practice, the tutorials themselves were fairly useless if you knew how to solve them
- it works very well
- see above ramble