

## DR CHRISTIAN URBAN

Compilers and Formal Languages (6CCS3CFL 2025/6 SEM1 000001) (6CCS3CFL-2025/6-SEM1-000001)

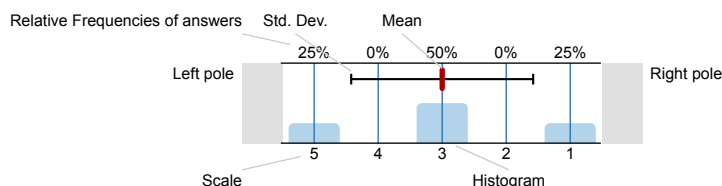
No. of responses = 51



## Survey Results

## Legend

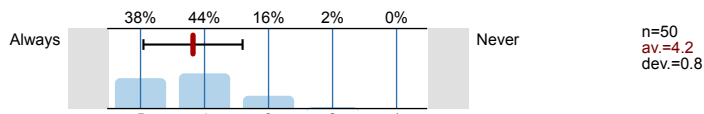
Question text



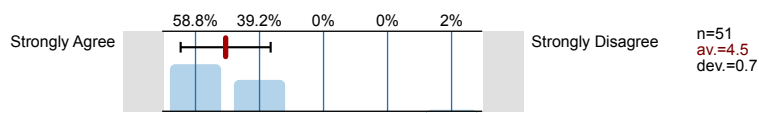
n=No. of responses  
av.=Mean  
dev.=Std. Dev.  
ab.=Abstention

## 1. Compilers and Formal Languages (CORE) - Compilers and Formal Languages (6CCS3CFL 2025/6 SEM1 000001)

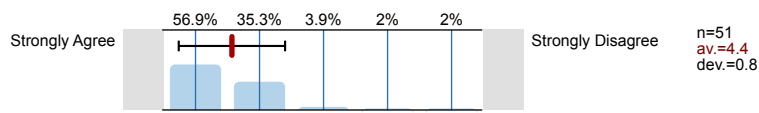
1.1) How often did you attend lectures, seminars, or tutorials for this module?



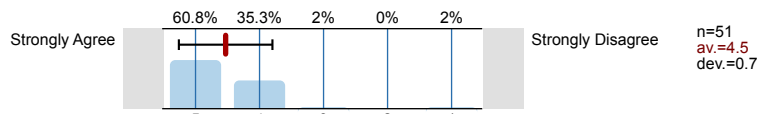
1.2) The module was well taught.



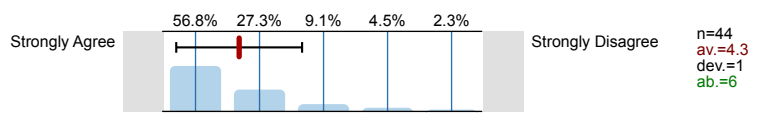
1.3) The content on this module was explained in a clear and accessible way.



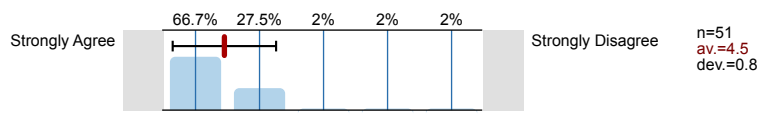
1.4) I felt supported in my learning.



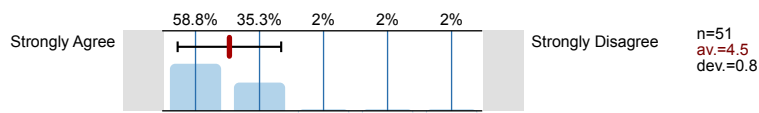
1.5) The feedback I received for improving my work was helpful.



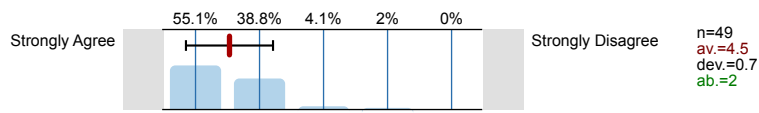
1.6) The subject-specific resources (e.g., readings, equipment, facilities, software) were easy to access when I needed them.



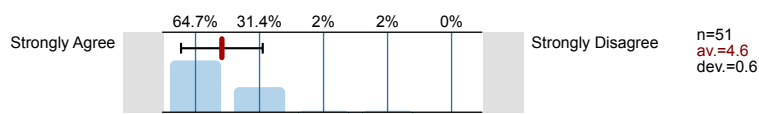
1.7) The module was well organised.



1.8) I felt included and encouraged to participate in this module.



1.9) Overall, I am satisfied with this module.



# Profile

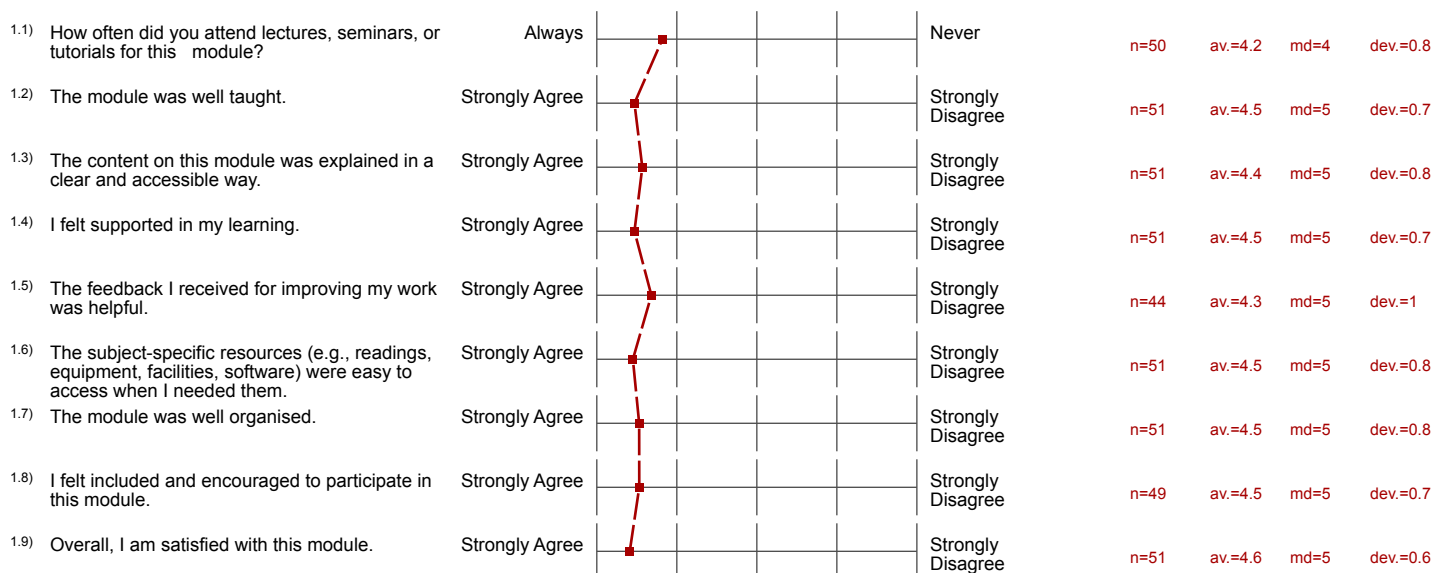
Subunit: Informatics

Responsible for modules: DR CHRISTIAN URBAN

Name of the course: Compilers and Formal Languages (6CCS3CFL 2025/6 SEM1 000001)  
(Name of the survey)

Values used in the profile line: Mean

## 1. Compilers and Formal Languages (CORE) - Compilers and Formal Languages (6CCS3CFL 2025/6 SEM1 000001)



# Comments Report

## 1. Compilers and Formal Languages (CORE) - Compilers and Formal Languages (6CCS3CFL 2025/6 SEM1 000001)

1.10) What have you enjoyed about the module?

- Christian teaches and explains very clearly. The coursework correlates to the lectures and helps understanding. In the SGT's the TA explains the answers to the questions and the questions are helpful in our understanding.
- Christian Urban is a great lecturer to listen to. His passion for the subject shows in the way he teaches it. When explaining theory, while sometimes a little long-winded, he's able to make confusing concepts understandable. My favourite part is definitely his handouts – I think they're vital for a confusing module like this and they almost always solve my questions.
- Christian Urban is by far one of my favourite professors during my bachelor's degree. He explains everything clearly and provides insightful feedback on any question that we have. He is extremely patient and explains the ideas in an ordered way while providing important background information, which makes us feel good because it is easy for us to connect everything we learned.
- Everything was great. Very engaging lecturer, and just an absolutely fascinating module. Loved every bit of it, the coursework, and the questions.
- I always thought creating my own programming language was a very difficult task. But after studying this course, I realized I could do it (even with some simplifications because of time constraints). This course laid a solid foundation for a deeper study of this field and exploring more complex aspects. I wish the Advanced Software Engineering program were filled with similar courses.
- I found the module intellectually challenging and it exposed me to perspectives on formal languages and theory that I had not previously studied in depth. While it did not align closely with what I personally enjoy most about compilers, it did broaden my understanding of the theoretical foundations that underpin certain aspects of language design and analysis.
- I like the lecture videos on KEATS, I think they are very engaging and I like the lecturer's sense of humour.
- It is one of the only modules where the lecturer is actually teaching in the LGTs and actively helping students. Videos are amazing, clear and to-the-point without just throwing content at your face.
- It was brilliant. The lectures were well taught, and the lecturer had very good humour.
- It was structured in a logical way chronologically of what a compiler does which made it a lot easier to follow the module content and understand it.
- office hours :)
- Overall, CFL was without a doubt the most interesting subject I have taken. This is due to the fact that I personally find the content, and coding very fun, and also because of the professor Christian who truly taught the module beautifully. I appreciate how passionate the professor is about the module, as that really makes the subject all the more wonderful. This module deserved all the nice lecture theatres we've had. I would like to say thank you.
- Really good module, opened up the world of Compilers for me, but I hope to go even more advanced.
- Really great teaching and explanations of the content! Thank you for this module!
- SGTs were very helpful and the TAs (i had a few due to absences) were very helpful and explained stuff very clearly when there was lack of understanding. Like PEP, the github test feedback also provided to be helpful at times.
- Thank you for delivering the content in such a clear way. The lecturer always explained everything thoroughly, making the module easier to understand!
- The difficulty of the CW was perfect.
- The hands on experience with programming compilers was very enjoyable. I especially appreciated the GitHub issues that automatically reviewed the latest push.
- the lecturer is enthusiastic and showed clear interest in the subject he is teaching. i like the git repository for the coursework and the feedback it gives it is really helpful.
- The lecturer is very passionate, friendly and approachable. His teaching style is great especially if you are caught up with the content, and then you can contribute, but the lectures are also good because they summarise the most important parts of the material. The worksheets and handouts are great as well.
- The LGTs - Christian is such a cool lecturer! Also the coursework has been enjoyable, although with a steep learning curve.
- The responsiveness of the professor. I'd often get an email reply from him within minutes. Something I valued a lot as it showed that he was actively looking out for students who needed some assistance and that helped me with my coursework greatly.
- The Small group tutorials with Flavio, I believe that's his name, were my favourite and engaging. Also asking Dr Urban questions.
- This was one of the strongest modules I have taken. Based on positive feedback from students in previous years, my expectations

were high, and the module met them. The lecturer's enthusiasm/passion for the subject was evident and made attending LGTs genuinely engaging.

The videos were particularly effective and explained the concepts clearly. While the module was not easy, it was appropriately challenging and intellectually stimulating. Some topics required time to fully absorb, but the level of support available made this manageable. The coursework was also a valuable exercise that reinforced understanding of the material. Overall, I thoroughly enjoyed the module :)

1.11) If you could recommend any improvements, what would they be?

- .
- Encourage people to start the coursework earlier if possible. I started early and it took me a while to get through it. I personally feel that starting the coursework on week 5 is a bit too late, maybe week 3 is better as we have exams coming up in January.
- hopping between different locations during the lgt's, though gave me an opportunity to see all the different locations. but not all of the sessions were recorded, do was detrimental to my studies
- It would be great to not be in a different lecture theatre each week, as this can lead to frequent late starts and disruption.
- It would be really helpful if the solutions to homework sheets could be released, especially for revision during the Christmas break. This would reduce the need to send frequent emails and wait for replies.

Please consider extending the SGTs, as not all questions are covered fully within the current time.

It would be beneficial to have dedicated lab sessions focused on coursework support — essentially sessions specifically aimed at helping with coursework-related queries.

Some of the lecture videos were a bit long at times, though they were engaging, so this wasn't really much of an issue.

- I understand Dr. Urban feels quite strongly about not releasing answers to the Homework questions. I do however feel this has a negative impact on learning. Relying solely on TAs in SGTs to provide feedback is an issue for many reasons. Some TAs might be unprepared, some might not be able to explain certain things well (despite knowing the answer), and overall individualised feedback isn't possible since there are only so many TAs on the teaching staff for a fairly large student cohort. In future, I believe it would be beneficial to students to have the answers, even if released with some delay, to support learning. This indeed might lead some students to attempt to memorise answers for the exam (which is what Dr Urban is attempting to avoid), but put frankly, those are the bad students and in general the exception, and I am a firm believer teaching should be tailored to those eager to learn, not to the ones who just try to get by with the wrong attitude. I don't think it is fair for students like myself, who put in the effort and are genuinely interested in the subject, trying to gain a deeper understanding of it, to be "punished" because of the ones who try to game the system.
- I wish this course had guest lectures from professionals who work on compilers, so that the course not only has a strong foundation in academic knowledge but also the ability to use the coursework as a portfolio project. It would be possible to somehow change the proofreading format, for example, by using input.txt and output.txt.
- Just minor improvements that could help:
  1. Timing of LLVM content
 

Introducing LLVM earlier in the term would be beneficial. Compared to earlier weeks, LLVM is a more substantial topic, and covering it in the final week felt rushed. While it is understandable that Week 8 content needs to precede LLVM, it may be worth reconsidering the placement of Week 9 (optimisations) so that LLVM can be introduced earlier (or maybe begin introduction to it in week 8 itself alongside the current content). Since LLVM is also required for CW05, earlier coverage would allow more time to absorb the material, work on the coursework, and ask questions during in-person LGTs or SGTs rather than relying solely on email/forum (but I do appreciate how extremely responsive lecturer is online!!). This would also help students complete the coursework before the holiday period :)
  2. KEATS typed exam format
 

This is more of a concern (that may never come true). I understand the motivation for a typed KEATS exam, particularly to avoid issues related to handwriting clarity (how it disadvantages people with bad handwriting, or students mixing up question numbers with corresponding answers). However, the module involves a significant amount of structure that is not always intuitive to express in ASCII, especially under time pressure. For example, automata are naturally drawn before being formalised as states and transition functions, and the same applies to ASTs or parse trees. Even for non-diagrammatical things like subscripts etc. I would've preferred writing them down because they're easier to follow for us as well when we are writing answers down. Although guidance on writing these structures in ASCII was provided in final LGT (and I 100% believe in lecturer to make marking focused on conceptual understanding), the additional cognitive load of encoding 'structured' writing textually during a timed exam is slightly concerning. The number of questions (23 in two hours) worries me for this, though it is difficult to judge fully without seeing the exam itself.
- Lecture content is simply far too long. I understand the need to explain things well, but there should be a limit and 2.5 hours certainly exceeds that. I believe a key reason for this is how many of the lecture videos are simply explaining (long) Scala code. When watching the lecture videos I'm primarily trying to focus on theory, and I have to search for it amongst Scala code explanations that I realistically don't understand until I'm actually working on the coursework. However, I understand the need and value for having these Scala code explanations. I can imagine the coursework would be even more challenging without it. Perhaps they should instead be kept separate – mandatory lecture videos solely focused on content, followed by another section of Scala videos for the week. That way, theory content can be learnt prior to LGTs, then Scala videos can be revised when the student is working on the coursework.
- More emphasis should have been placed on the coursework. The TAs and lecturer focus mainly on the content and worksheets, which is fair enough, but this kind of helps the "Out of sight, out of mind" situation quite a bit. It is 60% of the module after all. Additionally, though I do understand the reasons why solutions to the Homeworks are not given, not everyone could attend all labs and finish all worksheets before Christmass, so would it be possible for solutions to be released some time in Christmass?

- Please have the .pdf files for the lecture material be accessible via URL. I hate having to download them all the time!
- Sometimes, especially in the last few weeks, there seemed to be some inconsistencies between the content that was talked about in the lecture videos, lgts and sgts. For example, the week 9 sgt had some parts about compiler optimisations which is week 9 lecture video content despite sgts meant to be focusing on the previous week's content and the week 9 lgt had some llvm content towards the end despite that being week 10 content. I did not have much of an issue with following through with it but it did feel weird and confusing that the topic being talked about was not always the same between every taught session in the same week.
- the change of lecture rooms everyweek is not convinient
- The LGT's are a repetition of the lectures. It's not Christian's fault tho :)
- The module would benefit from clearer alignment between its title, stated learning outcomes, and what is ultimately emphasized in teaching and assessment. Based on both the coursework and the exam, the focus appears to lean more toward formal languages, particularly regular expressions and lexical theory, than toward compiler construction as a whole.

In a well-rounded compiler course, formal language theory usually serves as important background rather than the primary objective. The central aim is typically to help students understand the full compilation pipeline, including lexical analysis, parsing, semantic analysis, intermediate representations, optimization, and code generation. Assessments in such courses often reinforce this end-to-end understanding of how source programs are transformed into executable code.

In this module, the exam and the lexer component of the assessment place considerable emphasis on regular expressions and theoretical aspects of lexing. While this material is rigorous and valuable, its prominence in assessment suggests that theory plays a larger role than some students might expect in a compiler-focused course. As a result, other core areas of compiler design, particularly low-level code generation, receive comparatively less attention. Relying on the JVM for execution also limits opportunities to engage directly with target-level compilation concerns.

For me personally, this balance meant that the time invested in the module did not translate into the kind of practical, end-to-end understanding of compilers that I was hoping to gain. For students with similar expectations, this module may not be the best fit in its current form, and I would be cautious in recommending it as an introductory compiler course.

Overall, the module is clearly rigorous and contains worthwhile material, but clearer framing of its focus and a re-balancing of assessment could help it better align with the expectations of students looking for a more applied introduction to compiler construction.

- The SGT questions seem a bit unordered, containing information from weeks ahead and before. I also think it would be great to go through some example questions / problems in the LGTs, as before every LGT I went through the videos as recommended but sometimes the LGT would just repeat that information rather than test and challenge the information!
- The SGTs should be 3 hours long if I had it my way. Or 3 times a week. There should be large group tutorials also. The coursework should be a group project. Content should be slowed down or more time in the term. What do I know though.
- This might be controversial, but in my opinion having the CW deadline be all together in January makes things more difficult. I think I might have preferred it how it used to be, as I would have had to submit everything on time. Given the big deadline at the end, I focused on other things in the mean time and now have to cram :(((. But I do acknowledge that Christian gave a timeline for each CW and one would have just had to stick to that,
- While I understand Christian's position in regard to not publishing the homework solutions, I believe it would benefit the students who want to understand things than damage the overall course. The TAs are great, but especially during the first half of the module, we couldn't cover all the questions in the SGTs.
- Whilst the github test feedback could be helpful at times, it had its downfalls, especially as for most parts of the coursework it basically just told you if it compiles and or fails or not and then there would be 1 very small test and even so it wouldnt test your code for it. There would just be a screenshot or something which was very annoying trying to compare with own results as it would be lengthy at times.

Another big issue i had was not having access to SGT solutions or answers afterwards, whilst i understand that the exam content is supposed to be extremely similar to the content covered and you dont basically want to hand us the exam questions, i think this is unfair as people could have lots of reasons for not being able to attend sessions, or in my case when i revise later on and try to understand the questions i may forget what was said in the sgt and then i have no clear way to know if the answer i am doing is right or wrong. I dont see why we should be punished because the exam questions arent altered in a way that at this point they are near identical to the sgt questions