# School A Teacher 2 - 7 July 2016

Interviewer: Great.

Speaker 2: I said in the ... the introduction. But, I'll type up or, I don't know if I'll do it myself or if I'll get somebody to do, but I'll make a transcription.

Interviewer: Okay.

Speaker 2: And then I will send you guys the transcriptions and have a look at what you said. Because that's how research is supposed to work.

Interviewer: Yeah. I suppose.

Speaker 2: Is this what you remember saying? It is what you said, but is it what you meant?

Interviewer: That's why I decided to write lots of stuff down. At least, I make sure. As long as I make sure I get everything that I want to say.

Speaker 2: It's a semi-structured interview which is ... Feels a bit weird, because there are a list of questions. But, that means I get to ask all of the questions and it's far more of a conversation.

Interviewer: Yeah. It's just I don't want to mess anything up. I'm organised like that. I try to be, anyway. I always ... When the school news team interview, I do that.

Speaker 2: I know a bit about N–'s background, which is kind of varied. What did you do before this?

Interviewer: Nothing. This is it.

Speaker 2: You just can't begin-

Interviewer: Yeah. Yeah. I started off high school. Finished high school, went to the A levels. Did an A level in ICT. And then straight onto [inaudible 00:01:23] University. Or, University of Cumbria, now. Straight onto the three-year undergrad course, and then student teacher here. And then, got taken on a permanent full-time teacher. Been here ever since.

Speaker 2: Straight into computing? Have you always been-

Interviewer: I've always had a keen interest in computers, from an early age. Wasn't originally the career path I was gonna go down, teaching. In general. The background ... Before the computing came in, I did do a couple units, only very few units, on Visual Basic and things like that. So, coming in to all this, I got to learn a lot of this before delivering it.

Speaker 2: So, you've learned. You've taught yourself.

Interviewer: Yeah, self-taught. Loads of courses, loads of books. So, that's the thing that's been the biggest learning curve. Having to do it all myself first. Because the change was so big. I think that's probably why people are reluctant to take on computer science in some schools. Because it's such a big task to learn all the stuff yourself before, then do it. But, it's a new subject.

Speaker 2: Yeah. It's a completely new subject.

Interviewer: And, if you've not delivered. If you've got a background in IT, like I have, that is something that's completely-

Speaker 2: Yeah. It's very, very different. It's interesting to hear about high school.

I've definitely come across that at primary school, where the teachers really struggle to feel like they know what to cover. Or how to cover what they need to cover. It feels very foreign to them. Often. Even something like Scratch, which is pretty easy to pick up. I think it feels very alien to what they're used to.

Interviewer: Mm-hmm (affirmative)- I mean, Scratch is used a lot. Yeah. At a lot of primary schools. We find that they come in, the students, and say, "I'm here. I've used Scratch before primary school." But when you get down to it, they've not really done a great deal. And they don't understand what they've done. For them, it's just a building blocks exercise that does something nice. It's very colourful. There's a character, sprites on screen. It's the concepts behind it aren't covered.

We do a Pac-Man game. Or we did. We did our own variation on Pac-Man. And we start looking at that. And we start talking about using iterational leaping to get the keys work on the keyboard. And they don't even know how to get the characters moving on screen. So, what they do at primary school, I don't know.

When we have them and, for your five and six, we used to do Scratch. Then we were advised by county not to use Scratch as much because it's the experience that everyone's coming in with. That's when we started looking at Python, because it's a text-based programming language. We teach them the concepts, as well as the actual code itself.

We ditched Scratch. We'll probably ditch Scratch fully, eventually. I think it's one of them that people think they've used it, they think they know. They know everything there is to know.

Speaker 2: It's easier if you don't have that preconception. That they think they know what they're doing and they don't. It actually makes it a little bit harder, doesn't it?

Interviewer: Yeah, it does. Of course it does. Because you get people coming in with a misconception that they know all these things. And even some of the primary school teachers coming in saying, "Yeah. They've done loads on Scratch."

But, really, all they've done is actually create characters on Scratch. They've not done any of the programming challenges or anything like that. So, it is difficult.

Speaker 2: Even things, like talking about it being object-oriented, and what does that mean. And, how to loop things. And how to have different functions for different characters. It's often used just as a media service, really.

Interviewer: Yeah. We do refer back to Scratch when we discuss or talk through things like abstraction. And we're talking about things like: these are all the nice, little buttons or the nice, little blocks that you use. But, realistically, behind that there is a lot more going on. You'll see in this, this front end. There's not many other programming languages where you can do something and show something as big as that. There is the MIT app developer. But, again, it's unstable. It's not great, where Scratch they can relate to, because some have had experience at primary school who come in.

They aren't aware that there's so much that goes on behind those blocks. Until they start using a text-based programming language. And then you can refer back and say, well actually, all of this is what's going on behind Scratch while you're doing your loops on Scratch. And your if-thens. That's what's happening. So, that's quite useful. It's useful for that. Other than that, I don't think it's particularly useful anymore.

Speaker 2: The other thing I think that schools, primary schools, struggle to do, and this is, sometimes, networking. Taking advantage of the community. Because when Scratch is its most powerful, the kids look at what other people do.

Interviewer: Yeah.

Speaker 2: What I call deconstructive teaching rather than constructive teaching, where you build ... You take something apart, as opposed to putting it to together. But, often networks don't allow that. Or, the teachers are not aware that's a function.

Interviewer: Networks are always gonna be an issue in any school. Because of the nature of the subject, with what you potentially could do to a network with computer programmes. It only takes one student that's willing to go home and do a bit of research, and you could have serious network problems.

It took us, in here, a while to even get Google Chrome, because of security settings. And so we found out ... We went to another school that run Chrome and so we find out that Chrome adopts its security setting from IE. So, it was ... You could just put it on all the computers and it was fine. And that's the only way we could use App Inventor.

But then again, the emulator doesn't work in App Inventor particularly well. So, we've scrapped that idea. As a block-building exercise, a computer programming exercise, Scratch is the only one, really, that we've found that's useful enough to teach key concepts before going into fully text-based.

Speaker 2: So, coming back to the questions. Which I promise I won't stick to. How do you interpret the concept of computational thinking? And what do you see as the role of computational thinking, within the computing field?

Interviewer: As we do in every part of our curriculum, it's more taking problems and developing solutions to problems and wording them in a way that computers and humans understand. So, you're looking at things like abstraction, decomposition, pattern recognition, algorithms. That's the way we're seeing it going, at the moment.

Speaker 2: What do you think is the main purpose of ... I don't know how to say this. So, it's making it work so that computers and humans can understand one another?

Interviewer: Yeah.

Speaker 2: And, why do you think that's important in the curriculum?

Interviewer: This is a sore spot because I don't think it's relevant for everybody in the curriculum. I think that ICT still plays a major role in industry. I think we're focusing now, mainly, on computer science and programming and logical thinking. The logical thinking side behind it and the breaking problems down. Solve problems and solving those first is really useful as a general life skill. But, I think it's still very much subject-orientated. Subject-specific. It's only if you're interested a career in that particular area.

Speaker 2: So, you don't see computational thinking as necessarily a useful way of problem solving in lots of areas? It's a way of doing it for computers.

Interviewer: I do, yeah. I'd say more for computers. It does help you think logically and help you solve problems. But, I think, because of the nature of the problems that you would solve with a computer, I don't think you would benefit greatly from problems outside of there. You could still apply those skills to other problems outside. Practical problems. But, no, I don't think ... The relevance isn't there as much as I think people like to think that there is.

Speaker 2: Yeah. I think that's completely fair, actually. And I think, often, you get people from within the computing education world who are convinced that computational thinking is just the best thing.

Interviewer: Yeah. It is a good thing. And they will become really good problem solvers and things like that. But, I think that the problems they will be able to solve aren't problems that you would face all the time, in day-to-day life.

When they come out of [inaudible 00:10:02] industry or when they go on to further education or on to the workplace, because they're not. The skills to think logically, and think through things, and break things down into small problems is a good skill. But being able to do it in reality isn't always the same as doing something that you could do on a computer.

Speaker 2: Do you think that the over-focus on computational thinking means that it frames problems in a certain way?

Interviewer: What do you mean?

Speaker 2: Do you think it excludes other forms of problem solving?

Interviewer: Yeah, it does. I think it does. You can solve problems by not just breaking them down into little problems, and then solving little problems to solve the bigger picture. There's loads of different problem solving skills that you learn in maths. You learn in science, where you get hypotheses and you gotta prove or disprove it. I think that computational thinking is not the only problem solving skill, at all.

It's good. It's a very good problem-solving skill. But I don't think it's the only one out there.

Speaker 2: Yeah.

Interviewer: I think that the students will pick up skills from other subject areas. Mainly sciences, if I'm honest. Sciences are: computer science, your physics, your chemistry. They're the ones that you are more likely to solve problems in. Higher maths. And they teach them all sorts of different ways, in other subjects' areas.

Speaker 2: What do you think is the main purpose of the change from ICT to computing?

Interviewer: I think that the idea behind it is that we need more computer programmers for the new age of technology. That's what I think it is. We think that technology's gonna move so fast that we need to keep people coming in to the industry to keep things going. To keep things developing. And, to look to the future. But, at the same time, we're forgetting about one of the most important things.

Office workers and people like that are gonna need basic office skills. Or, the web developers that have no experience in web development until they get into college. And then, they're behind where they should be because they should have had experience in web development at key stage four.

Or, your animators. Because animation is more an ICT-based topic.

Speaker 2: So you feel like there's a whole load of things that have basically been dropped.

Interviewer: I think they've been dropped. With the assessing with levels thing I did with county. One of the things that I pushed for was to keep ICT. So things like looking at some test criteria and getting feedback on your work. That isn't necessarily a computer science type thing. A computer science type thing encourages you to test your work yourself and develop your solutions yourself. And, eventually, at the end, then you'll get somebody else to test it.

But, I think that the ICT side of ongoing development, of getting people to give you a little bit of feedback on your work and-

Speaker 2: Did you meet resistance to that?

Interviewer: On the ICT?

Speaker 2: Keeping some of those ICTs?

Interviewer: No, all the teachers were great. They completely agreed. Things like databases and things like that ... Just designing your basic flat file, simple and complex queries. They all seem to have gone from the computer studies curriculum. But, again, people are gonna need those things. And, we're sending people out into jobs now where we've been very specific on who we're sending out or what we're sending out of school. And, really, we're limiting their options when they come out. Because they don't have the skill set-

Speaker 2: There's an over-focus on that, kind of ironically, the IT industries. And the computing skills that they'll need for everything else are being lost.

Interviewer: Yeah. And I think, as well, it would waste student time. You'd have to look at this because I'm not big on this one anymore-

Speaker 2: You're allowed to have an opinion, it's okay.

Interviewer: They used to do, at colleges, if a student here did not opt to take ICT as [inaudible 00:14:04] C, they would do functional skill ICT at college to get them up to speed. And, if we're sending students out of here, arguably, at year nine without any major experience of basic office packages, or anything like that, they'll miss out on two years' experience and then have to do another year at college anyway.

Speaker 2: You feel it's almost doing a disservice?

Interviewer: Yeah. I do.

Speaker 2: Not teaching it. And that kind of follows up with that next question, which is: what do you think is the relevance of the computing curriculum to all the pupils you teach? In some ways, it sounds like it's not, in your opinion, it's not relevant.

Interviewer: No, it's not. It's like any other subjects. PE's not relevant for everybody and-

Speaker 2: But, there's an argument. You can make an argument with PE, for example. That everyone should know how to keep fit. And that those habits of keeping fit are instilled within this age. Do you think you can make similar arguments about programming?

Interviewer: I've struggled with that because I know, obviously, when you do get into ... When you've got the experience dealing with problem solving, and you've got the experience of even fixing computers and things yourself, there is the independence thing. Like, the independently keeping yourself fit from PE. But, I don't know. I really think that computer science is very ... It limits people's options when they come out of education. Because it's so specific and the skill set apart from, like we said before, the problem solving. Because the skills that they've developed are very specific, I think some may be wasted.

Well, that's because I come from an ICT background. And then, it's the bitterness of, well, let's just drop all of ICT. And I found it really, really useful. I'm not saying not to have computer science and just have ICT. I think there needs to be a balance there.

Speaker 2: You think that the curriculum, as it's written, is over-focused on computer science?

Interviewer: Yeah. It's massively over-focused on computer science, which I think, as I said. If it was half-and-half split-

Speaker 2: Do you think there's an over-focus on having ... creating IT professionals and entrepreneurs? And people who go off and become the next Steve Jobs and whatever? Or, it's just an over-focus on them having this certain way of problem solving for this certain industry?

Interviewer: I think the more we've looked into it recently, the more that the new GSSCs have been because of the new spec for the GSSCs coming out. I think the computational thinking is a huge part of it. Funnily enough, it looks like the examples of backtracks on being a very computer science-related subject to now going to being a computer studies-related subject, where you do pick up some ICT skills. But, I still don't think enough.

Looking at key stage four, [inaudible 00:16:56] requirements, we have to have ... Students have to have an experience of computer studies at key stage four as compulsory. So, that's why we do enterprise. We do the enterprise course.

Speaker 2: So, you fill in some back fills?

Interviewer: So, we back fill some of that stuff. But, now they've dropped the big GSSC for IT. And we're having to look for vocational courses for an ICT. So, we've got IGSSC for computer science. We're having to get a vocational course, just so people can still choose ICT. But, because students are limited with options, because maths and English are so big and people are trying to push for maths and English, they're getting extra time for maths and English, students can't take as many options as they used to.

Now, if I was a student, I would go for a GSSC over vocational course, which would mean that the vocational course in IT becomes irrelevant again.

Speaker 2: But, it's also interesting if you think about the way that ... That means that the computing skills are being framed, and that the focus on computer science means that computing is something that you do if GSSCs and A levels and go in the university. It's not something that, if you wanted, you couldn't get an apprenticeship in that work. I don't know. There used to be something. Network administration, which you don't necessarily need a degree course in order to know how to administer a network.

Interviewer: No.

Speaker 2: But the way it's being taught is forcing people to make that choice.

Interviewer: Yeah. I think it is. It's very closed door.

Speaker 2: You have to be a book learner, for example, to use basic terms. A book learner to kind of continue with computing, as opposed to, maybe, somebody who is more practically based.

Interviewer: Yeah. I think you close a lot of doors with computer science on where they can go in the future. Just because of the limit ... The skill set that they will have isn't relevant for certain areas. Like, say that you can pick up and you can learn on the job. You can get apprenticeships. But, it's getting your foot through the door at apprenticeships first. To be able to go on to the apprenticeships. To be able to go on to succeed in the apprenticeship. It's more training's gonna be needed in the beginning that shouldn't be needed because they should have learned the skills earlier on in their life.

Speaker 2: Yeah. It makes sense.

Interviewer: It just seems a waste to me. It does. Yeah. They've gone backwards.

Speaker 2: They think they've gone forward.

Interviewer: They think they've gone forwards and they've gone ... Everything's now is focused on ICT. So, we now need to drop some ICT. And, the industry is saying that there's no computer programmers coming out of school. So instead of going for the balance, we've just done the same thing. We've dropped IT and then computer sciences. In a few years, it will go full circle and the industry will say, "Well, we've got too many computer programmers." No one's getting jobs because there's too many of them out there.

Speaker 2: Or, the other thing that will happen is that all the people who could have gone into computer science will have been bored because they were in these maths classes with everybody. And you won't have any computer scientists or anybody-

Interviewer: I mean, you look at the size of the groups that we have here at GSSC. We're 15 and 20 students because they are your hardcore group of students who have a really keen interest in this. Not feeling like they have to do it because, I don't know, society thinks they need to do more programmers.

We still need people to work in offices. We still need people on payroll. I don't think we need to have this big push for computer scientists. I mean, don't get me wrong, I love delivering the computer sciences course-

Speaker 2: No, that's the interesting thing. You can tell that you-

Interviewer: At key stage three. I really like doing it.

Speaker 2: You are good at delivering it and you enjoy delivering it. You understand it. It's a subject that you personally enjoy. But, it's the impact. I mean it's that next question, which is: how do you think the pupils will use the skills and concepts that they've learned? And actually, it sounds like you don't feel like a lot of them will use them at all. Because they aren't going to be relevant to the lives that they'll be.

Interviewer: No. I think that being a problem solver is a skill you can learn, but if you are learning to problem solve in a subject that you have no interest in, I don't think that you will pick up as much as you think you're going to pick up.

Speaker 2: Yeah. So, in a way ... I have heard various people say, "Oh, it shouldn't be called computer science, computing. It should be called problem solving." Because the skills are ... They're similar skills. But, actually, if it was problem solving, it could be a lot broader.

Interviewer: Yeah.

Speaker 2: So that you could focus on using problem solving skills to solve lots of different sorts of problems that were relevant to the kids in lots of different ways. But, currently, because it's computer science, you're teaching problem solving with computers. And some of them will just say, "It's not relevant to me."

Interviewer: Yeah. "I don't like computers. I can't use them sort of things." And then, you get that reluctance to learn. They don't want to.

Speaker 2: And they resist learning the problem solving.

Interviewer: They don't want to do it. And then, they resist the problem solving, the learning of problem solving. And then, you've not achieved anything then. It's like any subject, if you've got somebody that isn't interested in keeping fit or ... They're gonna be resistant again to learn and pick up things in PE. It's always going to be the same.

I don't think that being a computer scientist or being a problem solver ... Some people can solve problems quite easily. And I think some people aren't meant to be problem solvers. It's one of them.

Me and N– have said that you go and look at ... You go to computer science fairs. You go to programming courses and things like that. And some people will spend all day just picking on the simplest things because they can't solve the problems. And some people will excel, just like in a lesson. And it's them people that excel at problem solving that you'd expect them to go to GSSC and these subjects because they can problem solve using programmes. Not problem solve-

Speaker 2: The thing that you're saying which, again, I think is really interesting is that because so much of the motivation of this was to create more people going into computer science.

Interviewer: Yeah.

Speaker 2: From what you're saying, you don't feel like it will do that.

Interviewer: No.

Speaker 2: Because the people who go into computer science will go into computer science anyways.

Interviewer: Yeah. Because they have an interest. And outside of skill, they will be the ones that go on-

Speaker 2: But the curriculum, as it stands, is not giving anybody a new interest.

Interviewer: Yeah.

Speaker 2: That they didn't have already.

Interviewer: No. I mean, they've tried to shoehorn things like app development in, which is fine. And students have a keen interest because mobile phones. They're all using apps. And, the game-making. You find a lot of students do games. But you think of how many people that come out of school, that school environment or college, and how many people actually, realistically will be able to get into games development or games testing. It's very slim. So they're trying to make it look more appealing, look more exciting.

But, the be-all end-all of it is: you've got some concept you need to learn and you need to apply them to this situation. And that's it. Be given a task, a computer science-related task, and be able to break that task down, solve all the bits yourself. And come up with some kind of solution.

Speaker 2: Interesting, as well. Because there's so much stuff about how important the games industry ... That digital economy. That's the rub. There's so much stuff at the moment saying how big it is now, how important it is. But in a way, it sounds like, actually, if you step back, it's overselling itself.

Interviewer: Yeah.

Speaker 2: Because it may be big and important, but it's not as big as important, that we need everybody going into the digital economy.

Interviewer: There's that many different areas of computer science you could go into that benefit others. Yeah, they're trying to put the fun spin on it. It's like a lesson, delivering a lesson. You try and make a lesson as fun as possible to keep people engaged. And I think they're trying to make the computer science curriculum as fun as possible and get all the stuff in there that they think the students are interested in to get interest. But, you'll often find that you'll capture the attention of a lot of students. They'll come to the GSSC and they'll struggle. Because they just don't have the skill. The problem solving skills.

I personally think it all boils down to: you can't or you can. And I've always thought it is definitely a subject that you either can or you can't do. And I've seen it for years, since we've been teaching it. There are some students that you can sit down and give them ... You could write the code for them. You could explain every single line for them. And they still will not get it. Because they just don't have that interest. Or that don't have that thinking ability to understand what's going on in front of them.

I don't think it's the same for everybody. I think it's very much if you're interested in it, if you've got the problem solving in the first place, it will help develop your problem solving skills. But I don't think that if you're naturally good at solving problems before in other subjects, I don't think it works too well.

And there'd be the students that you'll find struggling. And that's why we find that the top set at maths who are, should be ... Your lot more logical thinkers will excel more or pick things up quicker than your bottom set.

Speaker 2: But it also seems that, having watched the different sets, lesser with the bottom set, but definitely with the middle set, top set. Even within those sets, there are people who have interest and, also, the skill and people who don't.

Interviewer: Yeah.

Speaker 2: And people who it makes sense. Yes, there's more of them in the top set. But there are a few in the top set who it doesn't make sense to. Even though they're in the top set maths. And, then, there's also ones you see in the middle set. And, I think really, a handful in the bottom set classes as well, who it does make sense to.

Again, even though they're in whatever set maths, it makes sense to them. They can see it.

Interviewer: I don't think there's any real way of splitting the ability to get everybody. Other than, at GSSC, if you want to do the subject, you choose the subject, therefore you are gonna get 100% interest because they want to be there. There's no other way, other than grouping through maths sets.

Speaker 2: Makes sense, I think. It's one of those-

Interviewer: It's still flawed.

Speaker 2: It's a flawed prophecy. So what do you think is the most important thing you teach? In the computer lessons?

Interviewer: The problem solving. It's all back to the problem solving. All the time. I think that's the most important part of computer studies. It's solving problems. Being able to speak to other people. Being able to get feedback because we still deliver that, even though it's an ICT-based thing. Getting feedback from people. Developing your products. Developing things. That's the most important skill. There isn't any particular programme or part or subject that as important, as a whole.

Speaker 2: And even that problem solving you see as important, but not universally important.

Interviewer: No.

Speaker 2: What is the bit that you spend the most time teaching? And, also ... I can't decide how to interpret that question myself. Partly it's about just the thing that you spend the most time on. But, also, what is the thing that's hardest to teach?

Interviewer: The hardest thing to teach. Judging from what we've done, over the past few years, the hardest thing is the programming language. It is like learning a new language. Your French, your German. Because I think the only way to really concrete those skills and be able to get and know all the syntax errors and the syntax you need to have is by using it regular. And using it for one hour a week. And getting as much as you can in, for one hour a week.

If students go away and they don't have an interest and don't really want to go and do any more work on it, you are gonna fight a losing battle each week. Because they're coming in-

Speaker 2: And they don't get homework, do they?

Interviewer: They don't. We don't set homework compulsory, because of the nature of the subject. And, because a lot of the software packages are expensive.

Speaker 2: Yeah. Of course.

Interviewer: Occasionally, we will set them. But if we ever set them, they are just design-based, where they had to hand-draw something. Or, they write a set of instructions and bring them in. We've got to be really careful, because the software packages that we have to run [inaudible 00:29:28] subjects are very expensive.

Python programming is free. But it's also quite difficult to download.

Speaker 2: Yeah.

Interviewer: So, it's ... The only other option obviously, we have our IT rooms open all the time. After lunch, every morning. Every night at school. Just so that people can come in and, if they do need to use specific software packages, they can do. But, no.

Other than that, it is difficult. It's getting-

Speaker 2: So what do you think? Is that the thing that you spend the most time on?

Interviewer: Yeah. Just going back through code. All the time. Example code.

As I built up my skill set in computer programming, and I think probably most people do it, because it is the only way to do it, is: if I've got in front of me a problem I need to solve, I will not break it down into bits. I'll take one section and I'll go and have a look on how I can solve that. Normally, by looking at code that I've done previously. Or, by looking at code [inaudible 00:30:21]. And then, you'll solve problems by taking snippets of all the bits.

Speaker 2: Most of it. I've done some [inaudible 00:30:28] stuff and it's always ... find someone else's project that's almost the same and I could just about understand.

Interviewer: It's one of those. If it solves that problem, there's not really anything you can go and find. Because it won't solve that problem.

So, it is. It's like learning a language and you come into it over and over and over again. To get it-

Speaker 2: Do you feel like there's room to teach those skills of finding somebody else's solution?

Interviewer: The difficulty is: yes. I like the idea. Because that's what I would do. And that's what people do in the real world. No one in the real world would go "Right. Correct, arrest programme and stress away from the [inaudible 00:31:01]" to just put it all down on paper. They're gonna come across a problem. They're gonna have to do some research. The new computer science GSSC states that you cannot do any research during lesson.

Speaker 2: In GSSC. N– was saying that the exam was, the exam conditions of ... You can't do research-

Interviewer: You can't research. You can't talk to anybody. I understand that there's, obviously, the side of plagiarism, things like that. But it's ... You can't always problem solve everything on your own. You do have to have a system to problem solve. And-

Speaker 2: I was also thinking about the skills that you need for industry. I think you'd be hard-pressed to talk to professional programmers to determine whether being able to write something right the first time is more important. Rather, or, being able to work as a team, do the research. Being able to read somebody else's code-

Interviewer: Talk to other people. Get help from other people.

I can understand at GSSC. I wouldn't turn around to the person next to me and say, "Can you help me with this?" That, obviously, is plagiarism. But, to able to go on line and go and research, and find how they solve different problems. And then interpret that-

Speaker 2: I think of problems that are only available on the test.

Interviewer: Yeah. It's-

Speaker 2: Have a hundred different solutions that are there. Being able to find the one that's gonna work for you.

Interviewer: That's problem solving, that is ... The curriculum is based around problem solving. Solving problems all the time. But yet, if I had a problem at work, I would go speak to somebody about it. If you had a problem in life, you'd go and speak to somebody about it. Or you go and do a little bit of research about the problem and how to solve it.

They are taking that out of the GSSC. They're trying to teach people to be problem solvers. But I think there's too much emphasis on being independent problem solvers. That you can't talk to anybody else to solve a problem. You can't go and do any research to solve the problem. You have to do all this on your own.

Speaker 2: This is something that, again, feels like there's a real tension between professional programmers or computing in the real world and the way it's being taught ... You're being taught to be autonomous, responsible, and accountable for your work. And that's not reflecting how computers are used. Would you ...

Interviewer: I don't really know. That is a difficult question. I couldn't answer that really. You caught me off-guard with that one, I think.

Speaker 2: That's all right. It wasn't on the list. I think we've covered this.

But, other than adding in more ICT and taking out some of the language stuff, is there anything else you would change?

Interviewer: No. I'm in the subject for a reason. I really enjoy the subject. Other than just adding not to completely take over ICT, because I've grown to like computer science. And, I do believe that some of the ICT is outdated and old.

Speaker 2: Do you feel like there wasn't enough input from people who were teaching ICT on the development of this?

Interviewer: Yes. I think that's the biggest problem. I think when I've done this thing for county for the assessing without levels, that has been the biggest thing. Because we've been able to set our own boundaries. We've been able to dictate. We've obviously looked to the computing GSSC. We've looked to the new technical ICT skills vocational subjects.

And we've been able to backwards engineer them, so that we can then build assessment criteria from those. So, we're still working toward a set of national curriculum. We're still working towards what they need to get for GSSC, but because we've reverse-engineered everything, we can start from early on and build those skills into computer science GSSCs in IT.

Speaker 2: But it's like, now with the development of assessment criteria at a county level, that's when ICT teachers are beginning to have some input.

Interviewer: Yeah.

Speaker 2: And it would have been nice when they were actually developing it. If there'd been a little bit of a conversation-

Interviewer: Yeah. I think. We understand, obviously, ICT was a dying subject, in a way. But there were still a lot of skills that were relevant. And they didn't have to come under IT. I think that's why they backtracked. Some call it computer science and they started moving towards computer studies. Because computer studies would include ICT as well. So, I do think they did. I think they jumped in too quick.

Everyone needs to be a computer scientist, which is the wrong way to go about it. It's proven that, because they have backtracked. And they have put some of the IT elements back in there. I just think there still needs to be more.

We still do spreadsheets, but in our spreadsheets we look at computer science converting denary to binary, so we still feel the need. The stuff. That we've still gotta justify why we're doing a spreadsheet when, realistically, we should be able to go, "Well, no. If this student is gonna work in an office when they're older, they're gonna need this skill."

Speaker 2: They'll need to know how to create tabs and do formulas.

Interviewer: Exactly. Not that they can create a spreadsheet to convert from denary to binary.

Speaker 2: Beautiful graphs.

Interviewer: Yeah. It's one of those. It's still skills that you need. We did. And we still do a functional skills in IT which was okay, but we got ... We were harsh marked, because if you missed certain tasks in­­ the middle, if you don't follow everything by the book, you fail everything afterwards. So if you didn't print one piece of paper off when you should have done, if you miss that section at the beginning of the paper, even if you did the rest of it right, you fail. Even then, it was brutal, but it's an exam, of course. Functional skills in ICT where they pick up [inaudible 00:36:45].

At the moment, it's only vocational stuff that's coming out. I know the functional skills is vocation-able. Looking at the GSSC I'm teaching at the moment. We do animation, we do logo design. We do web development. We do a bit of database. We do spreadsheets. We do movie making. And I know the course is to fill some of those gaps, but not all of them.

Speaker 2: Yeah. Well, I think the other thing ... This is something that I see being part of computing education community is that there is a certain degree of feeling those are things that should be taught across the curriculum. So movie making should be ...

I saw a presentation by an organisation that promotes using movie making in English to tell stories. For all sorts of reasons. And that is a great idea, but it's not written into those curriculums. And it's not happening. Because those teachers have their own pressures. And they don't feel the need to-

Interviewer: Perfect example is private school. You look at private school, especially if it's an English specialist. They'll be looking to teach IT, but they'll want to teach English. And that's natural, is this. Cross-curricular is a very good idea. Digital literacy across the whole curriculum is a really good idea. But, in reality, if the staff don't have the skills, then it will just get put the side and it will never be covered.

Speaker 2: So, that cross-curricular teaching is a great idea and brilliant when it works. But also a brilliant way to pass the buck.

Interviewer: With me, literacy across the curriculum ... Even as computer studies teachers, we're expected to do the same.

My English isn't great. I'm good at the maths, but not with the English. So, I'm reluctant to do in bits of English with the students. It's natural, because if you can't grasp it and understand it yourself, you feel reluctant to then give false information, delivering to students. And, that would be the same with ICT.

Across the curriculum, because if people didn't understand the subject, if they're not trained in the subject, they'll be reluctant to do it. In case they get students that know more than the teacher. In case they get those awkward questions from students that the member of staff can't answer.

Speaker 2: There's so much about ... It comes up in computing. It's there in so much teaching. This worry about the students knowing more than the teacher.

Interviewer: I mean, the nature of this subject, it's one of the things you have to accept.

I'm quite happy to admit that we have now just got into year ten a student that will know more than me. That will know more than I will ever know about computer science.

Speaker 2: I think that's the interesting thing, actually. I think that it's really possible to have students who really do know more than you, but I think computer science is an area that really brings it to the front more.

Interviewer: Yeah, it does. I think any subject really. It's all about going away from school at the end of the day and doing that extra bit of research.

Speaker 2: But, I think it's also a bit about learning to teach, how to facilitate as much as teach.

Interviewer: Yeah.

Speaker 2: How to allow. Be someone who signposts and supports and teaches a process, rather than having to teach facts.

Interviewer: Yeah. Well, that's what we've had to do with a student. We were looking to put him through the GSSC at year eight because he knew. So we were facilitating him. We were giving him things to go away and read. Things to go away and do. Telling him different programming languages to learn. And he was quite happy then, going on with learning them. But, we would never assess him on that. Because we don't learn them ourselves. We couldn't assess those.

Speaker 2: Interesting. Yeah.

Interviewer: So, it was very much, he became a very, very independent learner. And, then, as a result, he will know more than we will. But it's the nature of the subject. So, if a student came to me with-

Speaker 2: I think you could have the same thing in English. You could get somebody who was a very good reader and was reading very quickly and read far more than the ... Not necessarily more than the teacher, but read well beyond their age group.

Interviewer: Yeah.

Speaker 2: And you could see how you could use the same process of allowing them. Facilitating them to think about books more deeply and read. But, I think it just isn't necessarily as obvious, possibly.

Interviewer: No. I think for a student to engage more ... My God, this is probably quite naïve of me, but with the sort of age we're in now, I think students are more willing to engage in computers and the mobiles and the tablets than they are with the book. That they're more likely to develop more skills on computers than they would from reading.

Speaker 2: This idea of teacher-facilitator and teaching facilitation. Teaching kids who are more advanced than the teachers, I think, is a really ... Quite a big one within digital education. And, I think that some of those questions about assessment are really important to that. Because how do you assess someone who has gone beyond? And it's not just gone beyond something. Done something that you've never done.

So, even if it's not a lower level. They've read a book that you've never read. They done maths that you've never done.

Interviewer: Or they've learnt a programming language that you've never took before. And that is-

Speaker 2: How do you assess that?

Interviewer: I mean you can only assess based on what they've done. If you would give them a project, you'd say, maybe, "Go away and learn Java. Come back and I want you to do this task." And then, within that task, you would assess their basic understanding of the programming language. Whether they've been efficient within their instructions. And, the project document, the planning stages, the task, then the development. All of those are still relevant, whether it's the same or a different programming language. I think once you've learned one programming language, you have the concept for other languages then.

But, that's the only way you be able to assess it. Other than you having to go learn it all yourself, which is a lot because you're only ever gonna use it for one student, ever.

Speaker 2: That's really interesting. I have to think about that.

How do you feel that you are judged and evaluated for your delivery? You know more about this than anyone, actually. Because you've just written assessment criteria.

Interviewer: Well, yeah. Looking at the assessment criteria, I think I've been on the ... I wasn't on the receiving end this time, which was strange because I helped write it. I helped write the stuff, so I'm giving it to other people, saying that this is how you will be judged and this is what you've got to teach to. And, the panic that come over them is they have to say, "How do we teach this stuff? What do we do? How do we plan it into our scheme of work? What levels are we supposed to be aiming the work at?"

And, the point is: all schools are different. Some schools will naturally have higher ability students and some will have lower ability. Your grammar schools will start further up the scale than your comprehensive schools would do. But it is that panic, to people. That they look and say: "This is what we'll be judged by. This is what we've got to follow."

But, I mentioned in my part of the whole presentation that it's their interpretation of those assessment criteria. Not that they need to be followed to the exact point. I don't know. If there a programming language, so they'll have to cover a text-based programming language. That could be any text-based programming language. It doesn't have to be Python in particular or anything like that.

Speaker 2: Do you think ... Did you put in examples within the assessment criteria?

Interviewer: No. No, we didn't. We put standardised material in the folder so that they could look at.

I went through an example of our key stage three curriculum that takes all of those assessment criteria. I had the opposite. I had my key stage three scheme of work planned based on what we found from the computer science, because we thought everything was going computer science. All the IT dropped. So, we planned based on computer science and obviously, the computer studies assessment criteria just had computer science with a bit of mix of IT. We just put some of the IT back in.

And then, we hit all the criteria. So, it was quite handy for me because I have written it. I've already done the scheme of work for it. And now people are asking for that scheme of work. They want the scheme of work. They want the buy the scheme of work because it fits perfectly within. Which shows me that not me, personally, but other people think that that is what they've got to follow.

Speaker 2: Yeah. That's what I was asking about. Examples. And I don't think it's a criticism. It shows the vacuum that exists. That whatever example is there ... So, if the example shows using Python, then-

Interviewer: They'll feel the need to use it.

Speaker 2: They'll feel the need. Even though it says text-based programming language.

Interviewer: Yeah.

Speaker 2: Probably again because of a lack of knowledge with ... I wouldn't even use the word like ignorant, it's just a lack of knowledge, within the profession. People don't know quite how many text-based languages there are. Which ones are easy? Which ones are hard? Which ones are-

Interviewer: The other thing you'll find are, from going to these meetings ... We're lucky here, it's only myself and N–, who are both trained in the subjects. But you'll often find that, because ICT or computer studies isn't a core subject, that you'll also get a lot of non-subject specialists teaching it who won't have the experience. Who won't know what all the assessment criteria mean.

They won't be able to understand. They won't know all the different text-based programming languages. So, they will be very tunnel-visioned. They'll be very: "These are the examples that somebody's given me, that is exactly what I've got to follow. And that's exactly what I've got to do."

It also depends on how many hours of IT they get per week.

Speaker 2: Yeah.

Interviewer: I talked to one school on the launch day of the assessing without levels. And they do a carousel of technology. They actually only get six weeks. So they've got groups of students for six weeks in a year and that's it. So they get six hours of IT-

Speaker 2: In the year.

Interviewer: Also computer science, in the year. Or, in a term. And, they'll swap round with the rest of the groups. Which isn't enough.

And if you've got someone, as well, as non-subject specialist-

Speaker 2: But that number of hours is not reflected in the assessment, is it? The content is reflected in the assessment, but nobody says you have to have this many hours of ICT.

Interviewer: No. They don't. Not at key stage three. No. At key stage four, you would have to fit the criteria for the GSSC.

Speaker 2: So, okay. Moving on the next question and I think there are more questions. I told you I wouldn't stay ... what do you think is ... This is this the one that you struggled with actually, but I'll be interested to see what you say.

What is your interpretation of the priorities from computing from N–, from A–, from H–. What do you think that people are looking for when they look at your teaching? And say, okay, you are teaching computing. What are they looking for? What do they want the kids to be learning?

Interviewer: I think it's strange. Because we know, we're in the subject area. N–'s expecting me to teach students how to problem solve. And, how to use text-based programming languages and how to teach the concepts of what actually goes on inside the computer. As opposed to using applications on a computer. Because he's a subject specialist.

Speaker 2: Yeah.

Interviewer: Andy, being the head, is a physics-trained teacher. It's me just assuming here, but I would assume, if I was going to see a computer science teacher teaching, that it would just be all code. It's just coding. That's all they do. Because that's what computer science is. And that's what people think of computer science is, when it isn't.

I often get parents coming in on openings and things. Not knowing the difference between computer science and ICT. The first thing that comes up is, oh, computer science is where you do coding. Is it? Well, that's part of. And I think that's what they expect.

Speaker 2: That's been part of the national conversation.

Interviewer: That's what people think computer science is. They think it's just coding. So, I don't know. I'll still look at what we're supposed to do at key stage four and look at the national curriculum guidelines and go from that. But, again, they may look at the assessment without levels criteria to see if we're hitting those. And, they will be more aware that it's a problem solving course. As well as a programme. As well as knowing what actually happens inside the computer.

I think you'd have to be a subject specialist to really know what-

Speaker 2: Do you think that's different to other subjects? Or do you think-

Interviewer: I think it's the same. Because chemistry. You're just dealing with chemicals. But, that's again. That's naïve. There's loads of parts of chemistry that you can cover. You have to be within that subject to know.

Last time, we were lucky enough ... I was lucky enough to be observed by the lead office of the inspector who was an ICT trend inspector. So, he knew what to expect. He knew what he was looking for.

Whereas, somebody else comes in, they're not looking at what you're delivering. They're looking at how you're delivering. And I don't think the content comes across as a real need. They don't look for the content, if that makes sense.

Again, personal opinion. If I didn't know the subject, I wouldn't know what I was looking for.

Speaker 2: Switching up for a second, because I'm a school donor as well. And, actually, if you think about it, in some ways the superstructure is that H– should assess the quality of teaching. And, then the quality of the learning is assessed by things like SATs.

Interviewer: Yeah.

Speaker 2: But it means that ... there's a lot in between. There's a lot in between the quality of the teaching ... You can be a great teacher and have kids who aren't learning. Or you can look like you're a great teacher, but the kids aren't engaged.

Those are two really blunt tools. And, you can see ... But, the structure is split like that, really. Because you don't have specialists. But, you also have exams that are very much focused on filling in boxes and meeting criteria.

Interviewer: Yeah. That's just it. It's meeting criteria that you got to meet.

Speaker 2: And that neither of those look at understanding.

Interviewer: No.

Speaker 2: Really-

Interviewer: But it's the argument that showing understanding, as well.

How do they show they can understand it? You can put grades on the board. You could say to a student: "Are you working towards grade five or grade six? Or A-star? B?" It's the understanding them, as what they're actually working. And how to get to the next grade. That's what they need to learn.

Speaker 2: It's one of the things I really liked about that lesson today. With those assessment criteria, they could be engaged in, at lots of different levels.

Interviewer: Yeah.

Speaker 2: The kid could put really basic things in and that would meet the criteria. But the ones who really showed the understanding would be the ones who said: "We need to do this because of this." And, each criteria had that reasoning behind it.

Interviewer: They're the ones you'll get more out of. But again, it's ... That's showing you. That demonstrates their understanding of what you're going through.

Speaker 2: Sorry ... I think the next question is: what's the hardest aspect of the computing curriculum to deliver in or to meet expectations? Of you, as a teacher. But, I feel like you covered that. In some ways, it's a mixture of teaching coding and at the same time, teaching computational thinking.

Interviewer: Yeah.

Speaker 2: But, really, the hard thing comes back to that coding. And it sounds like, in some ways, it's hard because it's hard. But it's also hard because it's hard to justify.

Interviewer: Yeah.

Speaker 2: It's hard to say: "This is really important." But, it does say they need to be familiar with ... Is it two or three? Multiple-

Interviewer: Programming languages. Yeah, it's more than one programming language. Scratch. It doesn't specifically say more than one text-based.

Speaker 2: One text-based.

Interviewer: At least one text-based. So, Scratch is one. And, but as I said at the beginning, if we drop Scratch, we'll have to pick something else. So, we're looking at App Inventions. MIT App Inventions.

Speaker 2: I suppose Microbit gives you a nice weight.

Interviewer: Microbit's good because, again, it's a block editor. And what we like about it, Microbit, is being able to teach that ... The concept of abstraction. Because you've got so many different elements to the software that comes with Microbit, so you go down to the block visual. Microsoft block editor, or whatever it was called. All the way up to, you got your Python, your Java. Actually, on the Microbit website, there's a sliding scale where you can actually see it going from the little blocks and the bits of code. And that's a really useful tool.

Speaker 2: I've seen that. What was the one? It was the Hour of Code, actually. I don't know if you've used any Hour of Code-

Interviewer: No.

Speaker 2: Resources. But, there's one ... If you ever have a kid whose really into Minecraft, there's an Hour of Code Minecraft thing. And it uses blocks, but you can also see it, I think, in Java. But, you write it in blocks. Then, there's a button you can press and see what it looks like-

Interviewer: I think that's really useful. Or, being able to do it the other way around. Doing the text and seeing what that will create as a block. I don't think there's anything out there that does that.

Speaker 2: Nothing that's that-

Interviewer: Well, that would be really useful. Because they could see ... They would then get that understanding that it works both ways.

Speaker 2: Yeah. That would be.

Interviewer: These software packages, these Scratch packages and things like that, that's how they would have been created. By doing it the other way around. But, they don't really experience doing the other way around. They get the experience of doing it one way and that's it.

Speaker 2: Mm-hmm (affirmative)- it is one of the big questions that comes up. Moving from blocks to text.

Interviewer: Yeah.

Speaker 2: If you were asked to observe and evaluate another teacher's delivery, what would be the key thing you would look for? And, I think you've actually done this.

Interviewer: Yeah. I've seen quite a few. The one thing I've still not seen is how it's the repetition in learning a new language. And the only way I can think of being able to do that is to, probably, go and see how language is actually taught. And how they get students to remember a lot of those ... I suppose it would be syntax. And it'd be your basic structure of bits of code. Like your basic sentence structures and how they remember how to do that in languages.

Because it's something new. For me, anyway. I picked it up because, some weeks, I was teaching it eight, nine times a week. So, I'm gonna pick it up.

Speaker 2: You're gonna use it a lot more. Absolutely.

Interviewer: If a student gets a syntax error or a problem, I can solve it straight away. But, it's how they can use it more and more and more without-

Speaker 2: More fluently.

Interviewer: Yeah. That's what I'd like to know. If I was to observe any other teacher. But, I do. I go and observe N– all the time. N– comes in and observes me, so I do have that experience of observing. And trained different ways of teaching, which is good.

Speaker 2: Yeah. And, there's a lot of back and forth in your teaching which is really obvious. Shows that you contribute in your teaching. And, also, your teaching styles are different, as well.

Interviewer: Yeah.

Speaker 2: Because if your teaching styles were very similar, there would be less value in that.

Interviewer: Yeah, we do. We have different teaching styles and students get ... We do try and split and try and change groups so in year eight, they might have N–. And then, year nine, they might have me so they get a taste of different teaching styles. Different way of teaching. But yet, still learning the same subject.

Speaker 2: Yeah.

Speaker 3: [inaudible 00:57:35] our work around the criteria for less than a month in computer research-

**Part 2**

….

Speaker 1: This is probably one of the hardest bits actually because it's about how the kids are relating to it so it's probably the hardest bit because it involves things that you don't know necessarily, which is fine but the other thing, part of the reason it's useful is that I will use, won't transcribe your answers right away but I'll use reflections from these interviews before I write the questions for the kids, hopefully. That's my excuse for not having done [crosstalk 00:00:33]. To what extent do you think the computing curriculum relates to how the kids use computers in their everyday lives, if at all?

Speaker 2: Yeah. What I was writing out, I put slim. Students use computers really to game, to communicate, social network. Very few students will go on, they'll run piece of place and code. I don't know. I don't think.

Speaker 1: Do you feel like the computing curriculum, the curriculum as you're delivering it is removed from all of those things? From gaming, from social networking-

Speaker 2: Yeah, the social network is falling of [inaudible 00:01:19], which is good, which is still highly relevant. The gaming I suppose to a certain extent but it's too focused. It's on your interpretation of what needs to be taught and how you're going to teach it. You could teach concepts of using loops and using selection statements by making your own games but I think it's harder to deliver those things making games than it would be just to do some solid coding. You spend more hours delivering it. Gaming to a certain extent. Communication not so much.

Speaker 1: Again, that comes back to the ICT thing. You think that there is more of that communication stuff in the ICT.

Speaker 2: There is, even just basics, the use of email and things like that and the social network, it's huge but nothing like ... It's covered in GCCICT, they even get asked in the exam about social networks and blogs and vlogs and so on but no, none of that, all that's going. When that's such a big thing, people get paid millions and millions of pounds per year to video blog on youtube, yet we're not gonna teach the kids about anything like that or the dangers of it. I know we have to do digital literacy and online safety-

Speaker 1: Isn't it interesting that there is a judgement then in the teaching community that some sort of jobs using computing are the right sort of jobs, making apps, programming software, being a web developer? Those are the right sort of jobs but being a video blogger who earns millions of pounds a year-

Speaker 2: Just on advertising.

Speaker 1: Just on advertising-

Speaker 2: Because they're popular.

Speaker 1: Just because you can make an entertaining video or you're a great musician or whatever. That's not the right kind of future or that's not one to be encouraged.

Speaker 2: No but it's what is gonna happen. There's nothing we can do about it because if you can make millions of pounds on just being popular.

Speaker 1: Just being funny.

Speaker 2: Yeah, being funny, being popular, getting people to like the stuff that you're doing, then ...

Speaker 1: The thing about the video bloggers and I don't know a huge number of them but some of them, it'd be hard to pin down what they do.

Speaker 2: Yeah other than sit at home and just video blog but it those [crosstalk 00:03:42].

Speaker 1: You were just talking about as well. It's the video editing skills and things like that, which-

Speaker 2: Which they don't have.

Speaker 1: Which these kids aren't learning, which again, isn't all behind the scenes with those video bloggers. Lots of them are very well, highly produced videos but smoothly done that you can't always tell.

Speaker 2: There's always, like I said, [inaudible 00:04:01], it's very much the computer science. I don't have a computer science background and I've taught myself so there is the possibility that if you've got that much of a keen interest, you will teach yourself and that is the case with ICT, if you're that interested, you'll teach yourself, which is fine for video blogging because you don't need a qualification to video blog. Whereas if you want to go into some kind of ICT background, you may need to do some kind of booster ECDL course to get the skills that you need to get a certificate, a piece of paper to say that you know how to do it even though you might already know how to do it because you've self taught. You look at my background and if I was applying for a computer science job, my background is in ICT, I may struggle but now I've got the experience and I've got the years of experience of doing [inaudible 00:04:47], it's different but you have to know it to get experience. It's a never ending cycle sort of thing, isn't it?

Speaker 1: Do you think the kids, to what degree do you think the pupils think the curriculum is relevant to their lives or not?

Speaker 2: Only if they have a keen interest in it. I don't think ... You'll often find that by the time you get to year nine, where they're thinking of their options, future, some of them will say, I'm not interested in computers, I'm not interested in doing this GCC because again, they're very naïve and they think that computer science is the same. It's all programming and that's it. They don't realise that it's actually problem solving skills. That could be down to the way we've delivered over the last couple of years that we have been very much python, project document, python, project document. Whereas now, we've started looking at things like sort bubble [inaudible 00:05:39] and insertion sorts and binary search and linear search so we're looking at the broader spectrum.

Speaker 1: Coding also is one of those things that is the least stable. When I was talking to you earlier, I learned visual basic and C++, neither of which are used anymore.

Speaker 2: No. I was set visual basic. That's what I learned. It does. Python was a big one all the sudden in skills and in a few years time, that might not be used anymore, might be learning java.

Speaker 1: Java or something new. There'll be a new one and this is one of the things I've talked to some of my other colleagues about, we talk about computing as if it's a science but everything about computing is constructed, including the languages. Somebody wrote that language and it behaves in certain ways because that person decided it should behave in certain ways and it deals with variables because that's how they thought it should. That doesn't mean that's the right way of dealing with variables.

Speaker 2: It's just their interpretation, it's their way.

Speaker 1: It's just somebody's interpretation of how and it works okay but when we're teaching it sometimes, we give this impression that that's fixed and that's reality. That's the only way for variables to be dealt with. Do you think that computing is gonna affect people's decisions and choices, I can't even read my own ... That one there. How do you think learning computing affects pupils decisions and choices about their future, again, if at all?

Speaker 2: The future within the subject or the future in general?

Speaker 1: In general. I mean, including about the subject but in general.

Speaker 2: As I said, with how difficult it can be, computer science, I'll do that one first. With how difficult it can be, computer science, [inaudible 00:07:31] because they don't see that broader it's actually problem solving, it's about breaking down these problems as well as coding or using code to break down the problems and solve them. That's what they don't realise. Even today, lots of them are looking at the c as cypher and encryption and decryption, that is problem solving and we're not using code there but it's getting students to understand that actually we've solved the problem. We've not had to use ... Yes, we're in computer science but I'm not through with python yet. I think that could be a mistake on our part again. It's that we're making them do ... We're making them take a problem that doesn't involve coding and adding code to it, which in a sense for them does put some of them off.

Speaker 1: Again, code is a very efficient way of doing cyphers. It's very fast.

Speaker 2: It is.

Speaker 1: Of all the things, that's just full blown, I think especially if you look at the coin flip versus the cypher one, it's hard to justify necessarily ... It's nothing against the programmer's work, it makes a lot of sense but it's hard to say why would you write a piece of code that would-

Speaker 2: To coin flip. That is more ... It's a box ticking exercise. It's making sure that those criteria that we have to meet are met. That's it.

Speaker 1: It does all the stuff they need to do and it's relatable to the lessons but with the cypher one, at least you think this is one of those things that people started learning code to do, was ... Computers were developed to do encryption. It's something computers are very good at.

Speaker 2: In terms of-

Speaker 1: It's used all the time today actually as well.

Speaker 2: In terms of, what was the other one, choice about the future, I wouldn't know. That's a really hard question to answer.

Speaker 1: [inaudible 00:09:27].

Speaker 2: About how it affects their future, I wouldn't know because we very rarely get any students coming back in and telling us news.

Speaker 1: It's also so new. This one is really-

Speaker 2: It is.

Speaker 1: Computing is a really hard one-

Speaker 2: We've not been running this long enough to know how it would affect the future, I don't think. I think you'd have to running longer for us to do it.

Speaker 1: Part of the thing about that question is that I feel like when somebody writes a curriculum for example, they have a certain idea of what that effect of that ... They have an idea of what the future will be like with that curriculum. It's come up a lot in this conversations actually, that there are some very clear outcomes that are assumed from the curriculum and those outcomes may not come about and it's interesting to explore whether those are happening or whether something actually different is happening.

Speaker 2: It is. Them in the future I don't know. I think it would probably be best asking them. All I know is that parents when we get the parents in, you always ... You always used to have ... Parents always used to come ICT, often at GCC, even though it's not based around ... It was very much if you wanted a job in ICT or an area of ICT, that you do the subjects and parents were under the misconception that every job needs ICT now and we always got that and we always found that we had to put people off and computer science parents now again, just think it's all down to code so again, that puts people off as well. I think in terms of affecting their future, we'd ask them. You'd have to ask the students but I don't know whether you get an answer to that now.

Speaker 1: What do you think or what do you hope, if I ask them in five years time, what they learned or what had the most impact, what do you think they'll say? What do you think but also what do you hope?

Speaker 2: I hope that they found the lesson full and engaging and that learning how to problem solve and learning how to think logically, or learn how to think logically and solve problems in different ways than what they'll learn across other subjects is what they would've kept. Other than that, if they've not gone onto a career in computer science, I can't imagine they would've kept anything, or used any thing. Because if I've learned computer science and I've gone onto be a firefighter, I'm not gonna use it. It is gonna go. I'm gonna forget it eventually because it's one of them things that you have to keep doing over and over and over again to stick with you. I think if you did speak to them in 19, 20 years, hopefully they will say that it was a fun way of learning how to solve problems and a different way of how to solve problems but if they've not used it, it wouldn't particularly matter. I wouldn't have thought.

Speaker 1: That's really interesting to think about as a fun way ... Reframing computer science or computing, as a fun way to solve problems. Also the idea, and I think you're right but I also think it's not there currently, that part of what we should teach with computing is how to solve problems in lots of different ways, that this is one way of ... That part of the process that you use for computational thinking but really for everything is looking at a problem and saying how do I decide how to solve this and what's gonna be the best way of solving this.

Speaker 2: That whole message. It's back to the original statement, is that it should be how to teach, how to problem solve in general and how to use different problem solving methods because at the moment it is a cross curricular thing because you will solve problems in other subjects and you [inaudible 00:13:26] subjects and solve things slightly different.

Speaker 1: You see glimmers of it like with the sorting lesson. That is very much that sort of idea, that you teach bubble sort and you teach quick sort, but you also can show that there are lots of different ways of sorting and they have different efficiencies.

Speaker 2: [inaudible 00:13:47]. It is definitely. It needs to be more how to problem solve and how to apply those problem solving skills as opposed to here's one way of problem solving, here's a programme and language, go and solve the problem using that programme language.

Speaker 1: It also sounds like and this is me trying to interpret what you've said so I might be wrong. It's for currently there are important aspects of problem solving that are being ignored because they're not computer-y enough.

Speaker 2: Yes.

Speaker 1: It should be about problem solving but there's not enough teamwork, there's not enough looking up and doing research. There's [crosstalk 00:14:30].

Speaker 2: If they're allowed at GCC, if they're not allowed to look up, that is a way of problem solving to go research, to ask other people to communicate but if they're not allowed to do it GCC, then we are very much blinking them, we're teaching them one way of problem solving and that is ... They only get the task there and then in the lesson so it's not even like they can go and do background research, which is another way of problem solving, do some background research and then ...

Speaker 1: Or go and think about it.

Speaker 2: Yeah. Here's a problem, go away, have a think about it for a couple of days and then come back. Another way of problem solving, to do that, the problem solving has got to be they've got to ...

Speaker 1: Answer a question.

Speaker 2: Answer a question there and then.

Speaker 1: Under pressure, under time pressure and not even be able to ... The other thing ... It's a problem management thing as well because so much of project management in computing is about going back to the client and this-

Speaker 2: Any job. Any job you think of, if you've got a problem, there's always somebody that you can go and see. Most of the time there is somebody higher up or there is somewhere where you can go. There's a computer system that you can use to-

Speaker 1: This is the other thing working with designers, part of problem solving, part of design thinking is the first part of problem solving is understanding the problem better and actually sometimes that means finding out that the problem you thought you were solving was different. That is part of that social skill again and you can't do that when you're given a test, a question. You can't go to the individual and be like what's this mean, is this what you really want.

Speaker 2: I think they're trying to get across the overarching problem solving skill is by breaking a problem down and that is, you look at any problem, the first thing you would do is you break a problem down with any kind of situation. I think that they've looked at this overarching skill of problem solving, which then is there's loads of other problem solving ways under this umbrella of breaking items down but I think we need to be looking at the different ways of breaking items down, not just looking at the bigger picture, being very blinking and not looking at all the elements that make up that big picture, if that makes sense. I think they do need to look at all the ways of solving problems.

Speaker 1: It's a very specific way of decomposition.

Speaker 2: It is but there's actually ... Students will see that and they'll see they won't make the link between what they're doing in here because we are the overarching skill problem solving skill and they will cover all the different, a lot of different subject areas that will probably cover all the different type skills. They'll cover the research skill to solve a problem, they'll cover the let's communicate with each other skills and go and ask somebody else and this, that, the other. They will cover all of those skills in other subjects but we're the umbrella over the top of that and they can't make the link to that. They're actually in most, if not all subjects, they're all about problem solving.

Speaker 1: Again, they're taught all the bits and pieces but it's not being drawn together-

Speaker 2: There's that connexion between everything that they don't have.

Speaker 1: That comes ... Thank you.

Speaker 2: This may have made-

Speaker 1: I think that's-

Speaker 2: It's made me think a bit more about it but the point of it, I think it's definitely ... In computer science, this is me being a bit naïve to what we do but it is very much breaking a problem down in its parts and solving each one of those parts by using different ways different methods so you might write algorithms for each different part, but still you are still decomposing a problem. You are breaking down its parts and that is only one way of solving a problem. There are many different ways but all the different ways that you would solve a problem, you'd still use the same method by breaking things down. If I come across a problem now with a computer, I'd look at all the different parts that were wrong with it and then I'd try and solve each bit first, if I had a problem with an appliance at home, I'd look at what part's wrong and then I'd solve each part bit by bit so it is still ... It's not [inaudible 00:18:53] sinking in some kind of overarching umbrella to problem solve that doesn't cover enough.

Speaker 1: I think there's a lot there. I think that you've picked up on a lot of the subtleties of the curriculum both what's good about it but also the problem with it and I think that there is something about problem solving and I think it's understanding what industry was really saying when it said there weren't enough people with computer science backgrounds. It also ... That was interpreted to mean they didn't have the right skills but it is also that there weren't people who were able to do the jobs because they couldn't understand the jobs in a way. Google doesn't necessarily want people who can programme in python. It wants people who can find creative solutions to the problems that its encountering.

Speaker 2: That's most of the time in business and business management, finding creative ways of marketing, finding creative ways of, I don't know, pricing strategies and things like that. You could argue that it does overarch a lot of stuff but there's also loads of things that make up that big umbrella of problem solving.

Speaker 1: Also, the other thing about it though is this idea of code and computer science and that within computing, there are actually lots of examples of teaching computational thinking without code and looking at computer science more carefully. All those unplugged activities that are out there do that in different ways and you could ... I was just thinking you could say to the students, I want to teach outside, what's the problem and how do we do it and make them think. Some of them might say we have to move all the desks outside or something but think, this is what I want to have happen, what are the problems involved and how do I get there, but at the moment because of some of the other aspects of the curriculum, the focus becomes on focusing on these things that are actually really hard for the kids that they work really hard at doing even though they probably won't ever feel successful with it.

Speaker 2: It's very much if they don't understand it, it is a switch off. I just don't understand it so I'm not gonna do it.

Speaker 1: I don't understand what a variable is.

Speaker 2: That's it so it's finding those fun different ways of teaching it to make it relevant to them and that's the only way I think is ... That's the same in any subject, I think with teaching any subject. I've taught math before and it was the same. You can have any subject that can be very dry, very boring. It's how you teach it that's the difference and would make the difference but even then, it's-

Speaker 1: Yeah but there's also something about the expectations and I think math ... One of the things I loved about studying the computer curriculum is because it's so new and we can see it in a perspective that we can't see math. We couldn't study how Shakespeare was put into the curriculum because it's been there forever and math is the same way but I think some of the power of that, with something like math is that it's been the same for a long time, it's evened itself out and now there are some teachers, not all teachers, but some teachers who make it really fun because they've had that chance to figure it out and it's been boiled down. Whereas with computing, I think we have a really interesting context that pushed it into the curriculum, with a series of expectations and because it's new, we can begin to think about, are there things that make it less fun that don't achieve what it's trying to achieve or what is it trying to achieve or why are we ... What's the purpose of teaching this? It is drastically different from ICT but that's not always recognised or the loss of that isn't always recognised.

Speaker 2: No. It is a move forward but at the same time, with moving forward we've also moved backwards or we will be moving backwards at some stage.

Speaker 1: Sideways.

Speaker 2: Yeah, sideways, because at some stage they'll turn around and the industry will go we don't have anybody that's got office based skills now.

Speaker 1: The accountants will start getting up in arms.

Speaker 2: Exactly. We're now struggling so now we're back tracking. It needs to be that healthy balance between the two. That's all it needs and I think ... I love both the subjects. I enjoy doing the ICT side of it, with the web development, with your animation. I love doing that sort of stuff but I also really enjoy the problem solving, the computer programming. I think the balance is just not there. The balance wasn't there when it was just ICT and not computer science. The balance isn't there now. It needs to go somewhere between so then we are churning out these students that have different skill sets because they want to have these different skill sets.

Speaker 1: This is again, something that I want to head towards is this question that what are the skills that pupils need for the future. That's a really hard question because it involves a little bit of crystal ball gazing.

Speaker 2: We don't know what industry is gonna-

Speaker 1: What is the future? You can do some guesswork. The thing about the computing curriculum is that these are the skills that they'll need but some of that is about this idea, I've just written digital economy in my notebook because what the digital economy is is far broader than just people who can write programmes. It's people who use digital tools to create value. That is animators and that is accountants. That's lots and lots of things and actually, we're not necessarily doing a service to all of that broad range of skills, broad range of professions by focusing on one area that happens to be one of the loudest voices.

Speaker 2: One of the things that we think is gonna take off the most but other things are still needed. The ICT side is still needed.

Speaker 1: And, this is the other thing, final thought for you to think about. We're losing some of the creativity bits. The other mixture to the digital economy is automation and artificial intelligence and robots and that stuff, which is getting very good and it will continue to get very good. If you read people who know a lot more than I do about this stuff, they say we need people who can do the stuff that algorithms can't, which is creative problem solving, not rote problem solving but creative problem solving, creativity and social skills. Those are the things that have been lost. Those very human, if we think about it, the very human aspects of using computers. Those are the really important skills.

Speaker 2: And the communication needed to use ICT, information communication technology. That's been lost as well. Interesting thought.

Speaker 1: There you go.