Pīkau Name: What is Programming?

Video Name: The Six Elements of Computer Programming (EMP07-5)

Presenter: Professor Tim Bell, Joanne Roberts

Joanne: Tim, I have noticed that in the computational thinking progress outcomes, it keeps referring to the six elements of programming which are, sequence, selection, iteration, oh I've gone blank.

Tim: Input, output and storage.

Joanne: There we go.

Tim: Yeah

Joanne: So why are these the six elements of programming?

Tim: Right, so basically with those six things you've got full control over any digital device. The reason that it's in the curriculum is that sometimes people concentrate on one more than the other, like you might spend all your time looking at input or something like that, and different ways of doing input, but if you never get onto something like iteration then students haven't really got full power over the device.

Joanne: How do we know that these are the six key elements? Why is it those six and not something else?

Tim: Ah, well they date back a long time actually. In fact, to a guy called Alan Turing, way back in the 1930's. He was around at the birth of computing and he is quite well known for things like cracking the German codes during World War II, and there is a movie about him called 'The Imitation Game' and so on. He did lots of cool things but he was thinking 'What is this computing thing we have developed? What's this device?' He came up with this idea of, well he called it an automatic machine, but actually it's become so famous we call it the the Turing machine - its named after him. It's just a mathematical sort of abstraction of what a computer is, but it turns out that although he wrote it in the 1930's, it basically applies to computers today. So we use that as a model of what computers can and can't do. The Turing machine, they don't really exist but basically it can be boiled down to those six elements. That was true in the 1930s of the very early computers, it's true in the 21st century of the computers that we are using today. When we make sure that students have

got access to all those six elements it means that we are giving them full power over being able to program a digital device rather than just concentrating on one area.

Joanne: That makes sense. I understand Input, Output, Sequence, Selection that kind of thing. Iteration which is repetition, why do we need that? Why does that need to be one of them? Can't we just write the same code over and over without using iteration?

Tim: Hmm, well you kind of could, so an example would be drawing a square, you know, draw one side and turn, draw a side, turn and if you do that 4 times you've drawn a square. You could write a program that's about 8 lines long to do that. Now it's a little shorter if you just say do that 4 times, but it's also more versatile. If you want to do something that has 8 sides then you would have to re-write the program and have 8 sets of commands in it, and then if you wanted, say, 360 sides and almost draw a circle or something, it becomes a very long program. So iteration, the ability to repeat things, the ability to somehow in the language, somewhere there's a way that tells the computer to do something over and over, actually gives you a lot of power of a computer. Now normally we don't use computers for drawing squares, but if you think of something like a digital camera, that's a computer, and it's adjusting the brightness of all the pixels. There is often a couple of million pixels, you know two megapixels, eight megapixels in a picture, and to change the brightness - it's digital, there is a number for every pixel, probably three numbers - you could write a program that goes through and adds something to each of those numbers to make every pixel brighter.

Joanne: Yip that would work.

Tim: That would work?

Joanne: Yip.

Tim: You could have a two million line program that says for the first pixel add this to it, for the second pixel add that to it and so on. Or you could write one loop that says "two million times: change the pixel value" and it goes through them all; and of course the other thing that makes it more versatile is tomorrow you might have a four million pixel image - four megapixel image - and you don't want to have to re-write the program and make it twice as long.

Joanne: Tim, when you talk about Turing complete, what does 'complete' mean?

Tim: Complete basically means that it covers everything that you'd want to do. When we were looking at the Bee-Bots, remember that we thought about how many commands you need to be able to make them move anywhere, and we realised that if you only had the forward and turn left command, that was complete. It enabled you to get anywhere on the grid. Now, it's not Turing complete, it can't do all the things that a Turing system could do, but its complete for the Bee-Bot's world in terms of moving on the grid. So forward and turn left, that's a complete set of commands, and if I add back and turn right I haven't changed the kind of things you can achieve, I've just kind of made it easier or more convenient or

something like that. But on the other hand you could have 30 commands for a little robot thing like turn left 90°, turn left 45°, turn left 180°, hundreds of turn left commands and hundreds of turn right commands, and even though it's got hundreds of commands, it's not complete because it can't get anywhere on the grid. So to be complete you only need a couple of things, and it's the same with the six Turing complete ones that appear in the curriculum. You can learn a few of them and you can achieve things with that, and it's good to scaffold the understanding, you don't necessarily want to teaching all six at the same time. Once students do get all six then they have sort of full control over things, but they might want to learn different ways of doing iteration, they will certainly want to see different ways of getting input to their programs, and things like that.

Joanne: So you have talked about Turing complete and about Bee-Bots not being complete, in what way are Bee-Bots not complete? What are they missing?

Tim: Bee-Bots can do sequence, one thing after another, and they have a form of output because you can tell where they have ended up - and they might land on numbers so they can output numbers or letters or something like that - but, for example, they don't have any way to do iteration, in that if you want something to happen five times or ten times you have to write every line of the program yourself and if tomorrow you wanted it to happen 20 times, go 20 steps forward, you have to write a new program. They really only have two of those six elements. Now in a Bee-Bot's world, which is just move around on a grid, they are complete. Using only two commands you can get anywhere on the grid, but they're not Turing complete, they're just sort of Bee-Bot complete, or grid complete. They can get anywhere on a grid, but to be Turing complete, in other words have full control over the typical things digital devices can do, they would need to be Turing complete.

Joanne: Do we need to use advanced languages to exercise all these six elements?

Tim: Actually a lot of the languages that students in primary schools are using are Turing complete. They cover everything that Turing said that these digital devices could do. If you think of a language like Scratch, it has input and output, it can obviously do sequences, the selection it has an "if" statement, and it can do iteration. There are a few ways of doing iteration in Scratch, but as long as there is one way, it is complete and it can store values, that's variables in Scratch. So languages like Scratch, and there are lots of them, they are complete, and I think what often goes wrong in school is that it's very easy to get the kids to experiment with it and they may look at lots of ways of output - you know, moving things across the screen or making a sound or things like that, it's one of the elements and its good to get to know that - but if they never get to look at the other aspects of it, they haven't really got the full power, the full control over what a device can do; but if you are teaching one of those languages it's absolutely fine.
I think that one thing about those languages is that it's a bit like training wheels on a bike. If you've got a young child and you give them the bike with training wheels on it helps them to get confidence and so on. If you're a bit older and you're using a bike with training wheels, sure it's a bike and sure I could ride one of those to work, but it just gets in the way. It doesn't stop it being a fully powerful bike that can do everything that I might want to do, but

at some point you prefer to use something that's maybe harder to use, but is able to do more of what you want it to do.