Code generation

Computer programming is being taught to kids as young as five. How will that help, asks Niall Firth

"ARGH, this is the most addicting game ever!" says Gabriel. "Do you want to try?" Gabriel is making a fruit machine video game with a twist. "Mine's an aeroplane game because I'm going to be a pilot," he says, spinning round in his chair before turning back to the screen.

At an after-school club in central London, six 9- and 10-year-olds are glued to their laptops. They manoeuvre coloured blocks of code and snap them into place, making bright cartoon figures dance.

Nearby, Imitiyaz has been having trouble with his fireworks. His rockets are launching, but instead of a sparkle of glitter, each one just turns into another rocket. Brow furrowed, he looks around for help.

But at Code Club, kids are encouraged to work things out for themselves. After a bit of trial and error he learns that he needs to copy in the explosion loops from another part of his program and snap them into a new place. "I fixed it!" he says with delight.

These kids aren't unusual. The only thing setting them apart from most school-age children is that they are getting a head start. This month, England is embarking on a big new experiment. Children going back to school over the next few weeks will find there's a new lesson on their timetable, as computer science replaces IT. Kids aged 5 and over will be taught how to program – and from the age of 11, they will be expected to get to grips with multiple coding idioms. "Modern languages" no longer means Italian, French or German. Computer programming will sit alongside reading, writing and arithmetic as the fourth core subject – a life skill for the 21st century. A generation of coders is on the way.

"Learning how to program is the new literacy," says Marina Bers of Tufts University in Medford, Massachusetts. But if that is the case, how does this new literacy differ from the old? And what will it mean for our relationship with machines? Surprisingly, there has been little research on the subject. There is some evidence that learning to code can boost the ability for abstract thinking and problem solving. And some think we are about to change our interaction with technology for good. Generally speaking, though, we are entering unknown territory.

Code Club was launched in 2012 by programmers Clare Sutcliffe and Linda Sandvik as a stop gap, to provide something that they felt children weren't getting at school. It has since set up nearly 2200 groups across the UK with volunteers teaching some 30,000 children the fundamentals of computer programming. Now coding is no longer just an extra-curricular activity, Code Club is branching out to offer training to teachers as well.

Designed with help from the Royal Academy of Engineering, Microsoft and Google, school computer science lessons will teach children how to build software, not simply how to use it. They will learn how to create simple algorithms and understand the logical principles that underpin coding.

The dramatic overhaul is the result of several years of lobbying by the technology sector – with the UK's video game industry at the fore. The Next Gen Skills campaign, for example, has been pushing for schools to teach the technical skills needed to develop the UK's digital economy. Demand for high-tech jobs continues to rise. "A major constraint on growth is a shortage of qualified graduates with the depth of knowledge that you get from having played around with computers for years," says Ebon Upton, co-creator of the Raspberry Pi, a $25 credit-card sized computer designed with the classroom in mind. ›

JON REINFURT

TOYS FOR CODING

"I want my kids to understand that computers are simple servants that can help them get what they want, not intimidating monstrosities that must be obeyed," says Dan Shapiro, who is on leave from Google to pursue his dream of teaching young children to code through toys. "My kids won't enter the workforce for more than 15 years. They're going to be living in a world surrounded by computers. The only question is, can they speak the language?" To make sure they can, Shapiro invented Robot Turtles, a board game that teaches algorithms. And it's just one of several such toys now available.

ROBOT TURTLES Age 4+
A board game in which you navigate a maze and capture jewels. To do so you must master fundamental coding principles, such as using a small pool of symbols to express complex ideas and planning a sequence before executing it.

PRIMO Age 3-7
A wooden puzzle with a twist. By placing blocks onto a board, you move a small wooden robot called Cubetto. Yellow blocks turn it right, blue ones turn it left, and red blocks make it trundle forward. The puzzle teaches the basics of sequencing and queues, fundamental to writing algorithms.

PLAY-I Age 5+
A simple graphical programming language lets you control the actions of loveable robots Bo and Yana. Older children can code more complicated instructions using Scratch. The robots teach concepts such as loops and if-then conditions, which make the robots behave in different ways depending on previous actions.

HELLO RUBY Age 4-7
In this book, a girl "with a big imagination" explores the world, tackling puzzles with helpful penguins and an array of other colourful characters. A companion book contains exercises that back up the story's subtle introduction to sequences, loops, if-then conditions and even open-source culture as they follow Ruby on her adventures.

But should all children learn to code? Clearly, only a few will end up doing it professionally. Not everyone needs to be a mechanic to drive a car – why should computers be different? Most day-to-day technology is designed precisely so that we don't need to know what makes it tick. Smartphones, apps, websites – they all just work. We have taught children how to use the web and Microsoft Office for years. Isn't that enough?

> "Children need to see technology as something they can control"

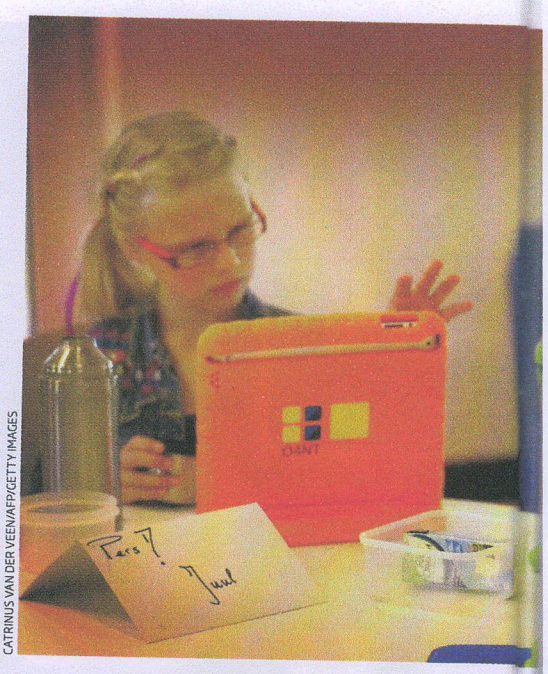
Not for people like Mitchel Resnick, director of the Lifelong Kindergarten group at the Massachusetts Institute of Technology's Media Lab. Coding gives you a new relationship with technology, he says. "It gives you a new way of thinking about yourself, a new way of seeing the world around you."

It's true, most children won't grow up to be jobbing programmers. Indeed, consumer tech – from smartphones to smart cars – now tends to sit in sealed boxes that we can't tinker with even if we wanted to. But, for Resnick, this is partly why a working knowledge of what's going on under the hood is so important. The skills children learn through coding will help them thrive in a world that is dominated by the growth of digital platforms. With technology now so dominant in our lives, children need to see it as something they can control, says Resnick. "Kids shouldn't just be the recipients of what others are creating."

Creative thinking

With that in mind, 10 years ago, Resnick and colleagues came up with a way to encourage children to get creating too. They developed Scratch, a programming language that is easy and fun for children to play around with. With Scratch, which the kids at Code Club use, you can create simple programs by moving ready-made blocks of instructions into the sequence you want instead of typing out programs character by character. You then run the code with the click of a button.

As students become more advanced, they can tinker more with the code blocks, and post their software on the Scratch website. Since the website launched in 2007, more than 6 million Scratch projects have been shared – all open to be reused, remixed and improved by others. "We want kids to design, create and express themselves," says Resnick.


CATRINUS VAN DER VEEN/AFP/GETTY IMAGES

Back at Code Club, Laura is steaming ahead. "We learned how to kill a witch. It turned into a ghost," she says proudly as she makes a photo of her own face spin onto the screen when a firework explodes.

So what is Laura learning? Some of the key concepts in Scratch have obvious parallels with more advanced programming languages. For example, a key part of coding is learning about sequencing: planning ahead to figure out the steps needed to make something work and the order they should come in. She's also learning about fundamental concepts like loops – snippets of code that repeat certain steps – and parallelism – the idea that sequences of instructions can run at the same time, even if they are doing different things.

Most important of all, though, may be learning about debugging: running back through some faulty code to see why it's not working as it should, tweaking it, then running the program again to see if that did the trick. Tinkering until something works is crucial in Scratch and programming in general. But it's also a key skill in life, says Resnick.

We learn maths and sciences not simply because they are important subjects in their own right, but because they train us to think in ways that are useful across the board. "Computer science is the epitome of that," says Mike Warriner, engineering director at Google UK. And working knowledge of how software can manipulate data could lead to

Teaching young children to code can give them powerful problem-solving skills

a more questioning mindset. "You're asking the same sort of questions as a scientist," says Amber Settle of DePaul University in Chicago. "But then you're asking how the data was processed too." In other words, learning to think like a computer scientist is about more than just knowing how a computer works.

However, there have been few studies looking at the tangible impact that learning to program has on young children. Most researchers still cite work done in the 1980s by Douglas Clements, now at the University at Buffalo in New York, showing that using the programming language Logo boosted young children's ability for abstract thought.

That makes sense, since computer science is built on the concept of abstraction. Just as a city map is an abstraction of actual streets and buildings – not to mention the people, traffic and general hubbub – so computer systems are constructed in layers, each hiding the details of the one below. When we hold a smartphone in our hands, for example, what's on the screen is the tip of an iceberg.

Abstraction is also important for designing algorithms, which are general procedures for solving many instances of the same problem. Knowing how to create an efficient algorithm requires an ability to distil a problem down to its essence and focus only on the aspects that you want to highlight. For example, a cooking recipe is an algorithm, where the ingredients and step-by-step procedure for making the dish are critical, but the utensils or kitchen you use are not.

Abstraction is one half of what Jeannette Wing of Microsoft Research in Redmond, Washington, calls "computational thinking". The other half is the ability to set out a problem and its proposed solution into clear, easily followed steps – something that a computer might carry out. Since coining the term in 2006, Wing's characterisation of computational thinking has been influential, even making its way into the design of England's new curriculum. The idea is that training children to think in this way – to focus only on the relevant points of a puzzle and step through them methodically – gives them a powerful problem-solving tool. It lets them turn big problems into smaller, more easily solved ones. "To reading, writing and arithmetic, we should add computational thinking to every child's analytical ability," says Wing.

"Scratch Jr is teaching kids the basics of coding before they can read or write"

But what does this all mean for children of 5? This year, toys are coming onto the market that are intended to teach even pre-schoolers the essentials of coding (see "Toys for coding", left). And Bers and her team have designed Scratch Jr, a simpler, purely graphical version of Scratch, to teach children as young as 5 and 6 the basics of coding – before they can even read or write.

Coding teaches children vital skills that teachers are trying to instil anyway, says Bers. Learning about sequences helps a child understand how to make a story flow from start to finish, put numbers in the right order, or have a better understanding of how a day's activities will proceed. "The idea that order matters is fundamental to literacy, to maths, to everyday life," says Bers.

Call to order

In one of the few recent studies to address the subject, she and her colleagues found that teaching children a simple programming language boosted their ability to tell a story in the right order.

Coding isn't the only way to teach the importance of ordering, but it brings the additional advantage that you can see the results of your actions immediately, Bers says. For example, if you program a robot to move four steps forward and see that it is going backwards instead, you can immediately tell that you used the wrong command.

Ultimately, we don't know exactly what impact computational thinking will have, says Settle. But she believes it will at least give a deeper understanding of how we get results from computational tools and thus how those tools can be tweaked to get different ones. That alone could make a huge difference.

For years, the big technological products we have come to take for granted, such as Facebook and Google, have mostly been designed by a very similar-looking section of society: white, middle-class men who went to the top universities in the US. Could educating a wider swathe of society in coding mean that the next Facebook, or the one after that, reflects more than just a narrow, engineer's view of how things should look or behave?

Bers hopes so. "I want future technologies to be created by a wider range of society, not just engineers and computer science majors," she says. "When anyone can create, we'll see things created by people with different mindsets."

Perhaps most important of all is that the next generation won't just be consumers of technology, but producers too, says Resnick. He thinks this will be the key knock-on effect of widespread coding know-how. "We will start to see people using computers for problems that until now no one thought was important," he says. "The whole of human society will benefit from that diversity." ■

Niall Firth is deputy news editor at *New Scientist*