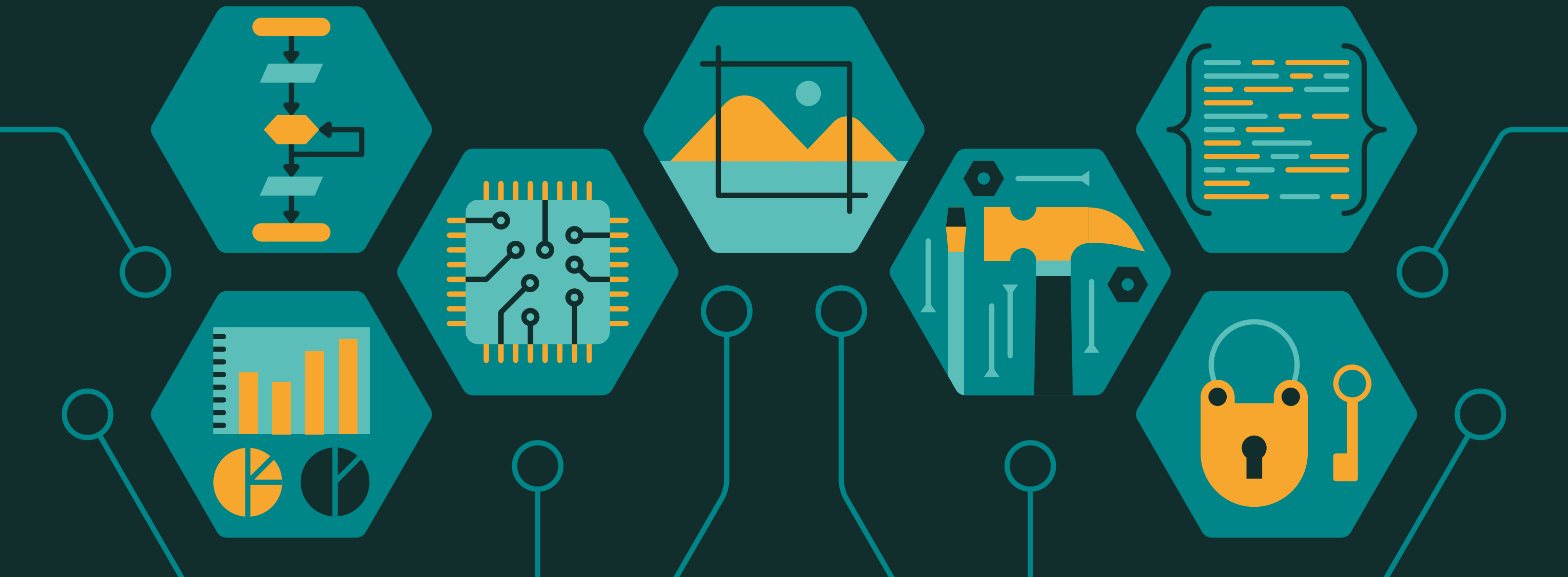
(Hello World)

THE BIG BOOK OF COMPUTING CONTENT



Raspberry Pi

Learn with the Raspberry Pi Foundation

Free for everyone anywhere in the world

Teaching resources

Discover training, resources, and guidance to help you teach computing with confidence.

teachcomputing.org

Project library

Browse our 200+ online project guides that include step-by-step instructions for learners.

projects.raspberrypi.org

Digital making at home

Check out our code-along videos and take part in Astro Pi Mission Zero from home.

raspberrypi.org/learn

Support for parents

Watch our support tutorials and access engaging resources for your child.

raspberrypi.org/learn

HELLO, WORLD!

After the warm reception our first special edition, *The Big Book of Computing Pedagogy*, received, we're pleased to welcome you to our second special edition, *The Big Book of Computing Content*. While the first book focused on how we can teach computing, this new book is about what we can teach within the discipline and really demonstrates the wide applications of this constantly evolving subject.

We have structured *The Big Book of Computing Content* around the Raspberry Pi Foundation's framework for formal computing education. This framework was originally developed by the Foundation for the Teach Computing Curriculum, as part of England's National Centre for Computing Education, and comprises eleven strands, each consisting of a range of conceptual and applied knowledge outcomes. It aims to categorise computing content to both demonstrate the breadth of computing as a discipline and to provide a common language to describe the different competencies and areas of study.

This book complements our first special edition; as such, it follows the same principle of introducing you to up-to-date research followed by our favourite stories, from past Hello World issues, of educators putting a concept into practice. For each of the strands, you'll find a table of learning outcomes illustrative of the kinds of knowledge and understanding that learners could develop at each stage of their formal computing education, from ages 5 to 19.

The second album is always a difficult one, so we'd love to hear your thoughts about this edition! Last time, we heard of educators setting up *Big Book* reading groups, of leaders using the book to support pre-service teachers, and even of a translation into Thai! Please get in touch to let us know how you're using *The Big Book of Computing Content* at **contact@helloworld.cc** or on Twitter at **@HelloWorld_Edu**. Happy reading, happy learning!

Gemma Coleman
Editor

EDITORIAL

Editor
Gemma Coleman

Subeditors
Louise Richmond and Amy Rutter

Subscriptions
Joshua Crossman

Social media
Neena Patel

Pedagogical consultant
James Robinson

DESIGN

criticalmedia.co.uk

Head of design
Lee Allen

Designers
Ty Logan, Sam Ribbits, Olivia Mitchell

Photography
Raspberry Pi Foundation, Adobe Stock

Graphics
Rob Jervis

Cover and illustrations
© Muti, Folio Art

CONTRIBUTORS

James Abela, Chris Aviles, Pete Bell, George Boukeas, Mac Bowley, Claire Buckler, Andy Bush, Katharine Childs, Jon Chippindall, Helen Clothier, Gemma Coleman, Josh Crossman, Paul Curzon, Diane Dowling, Emily Dreimann, Catherine Elliott, Lucia Flóriánová, Ben Garside, Ged Gast, Sway Grantham, Andreas Grillenberger, Hannah Hagon, Ben Hall, Tony Harkins, Matt Hogan, Zach Huffman, Michael Jones, Peter Kemp, Eirini Kolaiti, Thom Kunkeler, Doug Lloyd, Duncan Maidens, David J. Malan, Alan Merrett, Matthew Moore, Martin O'Hanlon, Matthew Parry, Rob Parker, Simon Peyton Jones, Carrie Anne Philbin, Pam Popay, Paul Powell, Oliver Quinlan, James Robinson, Harriet Ryder, Stefan Seegerer, Sue Sentance, Charlotte Spenceley, Jane Waite, Matthew Wimpenny-Smith, Andreas J. C. Woitzik

Hello World is published by the Raspberry Pi Foundation

Contributing Partner

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

INTRODUCTION

Supporting educators in providing high-quality computing education has always been integral to the mission of the Raspberry Pi Foundation. In 2018, having delivered in-person training to over 2000 educators, our content and curriculum team began developing more curriculum resources. The UK government had recently announced significant future investment in supporting computing educators as we began a piece of work to help us describe the subject of computing and in particular, the common threads running through it.

At the time, schools in England were offering the relatively new national curriculum subject of computing for learners aged 5–14, followed by elective qualifications in computer science. While exam specifications typically provided detailed learning outcomes categorised into several areas of study, England's national curriculum for computing consisted only of 25 statements outlining expectations for learners.

When describing computing, it was common to divide the subject into three areas: information technology, computer science, and digital literacy. Although this went some way towards outlining computing as a discipline, it was our view that this model created an artificial divide between aspects of the subject that were seen as more or less technical. Our more holistic view of computing recognised that concepts and skills within computing were far more interconnected.

There are, of course, many existing approaches to dividing up the subject matter, in the form of exam specifications, textbooks, schemes of learning, and various progression guides. We reviewed examples of these from England and beyond, and decided on some principles for our organisational structure:

- It should represent the whole of computing, incorporating what had commonly been categorised as computer science, information technology, and digital literacy
- Our structure should be capable of being applied across phases — the knowledge encountered by five-year-olds should be categorised using the same model as that used for our oldest learners
- The structure would initially focus on the English national curriculum, but would be independent of any particular exam specification, and would be capable of adaptation to new curricula in the future
- Computing as a discipline is a broad mixture of concepts and skills that need to be represented in any structure

Following these principles, we initially identified ten themes or strands that threaded through a learner's journey in computing education (see the diagram on the next page). This representation of the knowledge and skills that make up computing became known as our computing taxonomy. This taxonomy subsequently became a cornerstone of the work of the National Centre for Computing Education in England (of which the Raspberry Pi Foundation is a consortium partner), providing a common language to describe computing.

Given the interconnected nature of computing, we embrace a best-fit approach to content categorisation, choosing the most appropriate strand or strands for each idea. In developing our content, we identified that four of the strands were best taught throughout the others, in context rather than as discrete topics (the horizontal strands in the diagram on the next page). An example is 'Safety and security', which focuses on supporting learners to realise the benefits of technology without putting themselves at risk. While this could be taught as one discrete experience, revisiting this strand throughout a learner's journey provides regular reinforcement, as well as grounding it in the context of other strands.

Computing is, of course, a constantly evolving field and as such, our taxonomy will evolve with it. Since 2018 we've developed our taxonomy to incorporate what we've learnt, such as our response to the rapid emergence of artificial intelligence (AI) in recent years. AI represents a significant area of study and so is now represented as its own strand in our current taxonomy.

What we present in this book represents our work so far in describing the diverse range of concepts and skills within computing. We hope this book, and our taxonomy framework, resonates with your teaching. We welcome your feedback and contributions to help us develop this model further.

HOW WE ORGANISE COMPUTING KNOWLEDGE

Formal education content produced by the Raspberry Pi Foundation is mapped to the following strands of computing concepts and skills

- COMPUTING SYSTEMS
- NETWORKS
- CREATING MEDIA
- ALGORITHMS AND DATA STRUCTURES
- PROGRAMMING
- DATA AND INFORMATION
- ARTIFICIAL INTELLIGENCE

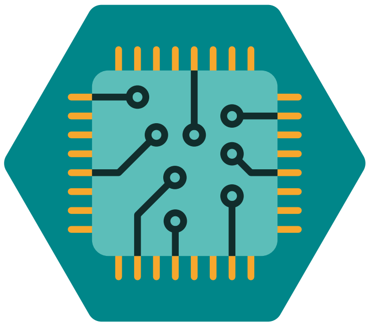
- IMPACT OF TECHNOLOGY
- DESIGN AND DEVELOPMENT
- SAFETY AND SECURITY
- EFFECTIVE USE OF TOOLS

CONTENTS

COMPUTING SYSTEMS

The study of computing systems is all about understanding what a computer is and how its constituent parts work together as a whole. A solid grasp of this concept is absolutely essential for any computing student; understanding how a computer system works allows learners to write instructions for computers, as well as to create artefacts with them. Knowing how a system stores images in memory, for example, might influence how students create and save images for a specific purpose.

Within this concept, students will focus on hardware and software, and the further they explore these themes, the more granular their focus will become. With hardware, for example, learners will gradually shift their focus from thinking about the system as a whole, to the individual devices, to how individual components (for example, the central processing unit or memory) work, and eventually, to the physical processes on which the system is built. Similarly, learners will typically begin by unpacking the difference between hardware and software, before coming to understand the different types of software, and finally, how hardware and software work together in a computer system.



IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** computing systems, in summary
- **What the research says:** what is a computer?
- The input-process-output model
- Hands-on Boolean logic gates
- Logic gates in Minecraft
- Introducing the world of quantum information

IN SUMMARY

COMPUTING SYSTEMS

Understand what a computer is, and how its constituent parts function together as a whole

STAGE 1	STAGE 2
Identify familiar examples of computing systems in the local environment	Examine how computers process input data in order to produce outputs (IPO)
Name types of computing device, such as laptop, desktop, tablet, etc.	**Explain that computers require input to perform a task**
Recognise key features of computing systems	Explain that computers produce useful output from a task
Describe the features and uses of different computing systems	**Identify the inputs, processes, and outputs of specific computing systems**
Explain some capabilities and limitations of computing systems	Classify a broad range of input/output devices
	Identify the capabilities and limitations of individual computing devices and components
	Write programs that use the inputs and outputs of a physical computing device

In the table below, you will find learning outcomes associated with the 'Computing systems' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
■ Describe an extended IPO model that includes storage (IPOS)	■ Distinguish between different types of software such as application, utility, and system	■ Explain the structure and operation of the main computer architectures
■ **Compare general-purpose, purpose-built, and embedded devices**	■ **Describe each stage of the fetch-decode-execute cycle**	■ **Describe the function, application, and principle of operation of specialist hardware and software**
■ Identify the purpose of a computing system's core internal components (CPU, memory, storage)	■ Compare different types of memory and storage including cache, RAM, and secondary storage	■ Explain the role of registers and buses in the fetch-decode-execute cycle
■ **Define an operating system and summarise its role**	■ **Describe different methods and media used to store data**	■ **Explain the role of the BIOS in configuring and booting hardware**
■ Identify simple logic gates (AND, OR, NOT)	■ Describe the functions performed by the operating system	■ Explain how resourcing and scheduling are managed by the operating system
■ **Convert between binary and denary representations of numbers**	■ **Explain the function of internal CPU components and their effect on performance**	■ **Create and convert between Boolean expressions, circuit diagrams, and truth tables**
■ Explain the purpose of compression and provide examples of its use	■ Identify common logic gates and circuit diagrams and explain their function	■ Apply the rules of Boolean algebra to manipulate logical expressions
■ **Build and program systems using physical computing components such as buttons, LEDs, and sensors**	■ **Construct truth tables and simple Boolean expressions to represent logic**	■ **Apply bitwise operations on pairs of binary numbers**
	■ Solve problems by using calculations and conversions with binary, denary, and hexadecimal numbers	■ Explain the difference between high- and low-level languages and the need for program translators
	■ **Describe how some common compression techniques work**	
	■ Build and program physical computing systems that exhibit autonomous behaviour, such as robots, control systems, etc.	

WHERE ARE ALL THE COMPUTING SYSTEMS?

STORY BY Sway Grantham

When I first started teaching programming and algorithms to five-year-olds, I began by using Bee-Bots, small floor robots you can program by pressing directional arrows on their backs. The theory behind using such devices is that they are more concrete than programming on-screen, and therefore more accessible for younger learners. However, I kept coming unstuck. How do learners connect what they're doing on a Bee-Bot with the computer systems all around them? Here is the research I read and the tasks I tried on my quest to find an answer!

Manipulatives

Let's take a minute to go back to some of our favourite learning theorists: Jean Piaget and Jerome Bruner (**helloworld.cc/piaget1952** and **helloworld.cc/bruner1964**). Piaget believed learners couldn't even begin abstract thinking until they were eleven, and Bruner recognised that learners needed to do repeated actions first (action-based thinking) before they could represent those actions on paper (image-based thinking). Both theorists support the idea that we need to work on a learner's concrete understanding and that, as a learner progresses, they will transfer this to more abstract contexts.

This application of learning theory supports what many educators have found when using manipulatives such as Bee-Bots. For example, researchers Sapounidis and Demetriadis conducted a study to compare a tangible user interface for controlling a robot with a graphical user interface (**helloworld.cc/sapounidis2013**). In interviews with the participating children, they initially preferred the tangible, suggesting that it seemed more fun and engaging. Younger children got on better with the tangible system, although this could be more to do with their developing mouse-control skills. Other research on physical computing also finds increased engagement with hands-on tools, and greater problem-solving skills, so there is definitely support for this approach — but this is where things started to unravel for me.

I found that learners could explain what an algorithm was, and that a program was 'a set of instructions that runs on a computer to tell it what to do'. Both met the curriculum needs, but I wasn't convinced they could link these two facts together. Could they connect what they were doing on the Bee-Bot to the computing systems around them? Did they understand what a computer was?

What is a computer?

According to my class of nine- to eleven-year-olds, a computer is:

- A piece of technology
- A keyboard and a screen
- A search engine
- A machine used for work
- A metal brain
- A machine with a keyboard
- An information device
- Electric

This simple question highlighted a wealth of alternate conceptions about programming and computing systems. Many children identified that a computer needed a keyboard. Many also believed that the terms 'machine', 'technology', 'electrical device', and 'computer' were all synonyms. The other commonality was describing the computer's function, as if we just need to know what it does to define it. This view of a definition leads to a reduced understanding of what computers are capable of.

Here's a useful activity to explore this question with younger children. First, get a piece of paper folded into quarters. In the first quarter, learners have two minutes to draw a picture of a computer. Nearly all of them will draw a laptop. Discuss what they drew — did their laptops include a keyboard and a mouse? What about a screen? By acknowledging the parts of a computer, you can later explore which parts are necessary for a computer to work. Now move on to the second quarter. This time, ask learners to draw a different type of computer; you will usually get a mixture of desktop computers or games consoles connected to a TV. Again, talk about the parts. Now you can have a discussion about there being no keyboard on a games console. Repeat this process, but change the question to 'What objects do you think have a computer inside them?' Each drawing they do leads to interesting discussions, from traffic lights, to remote control cars, to iPads.

My learners now had two discrete chunks of knowledge: how to program a Bee-Bot, and that laptops were computers. However, without a bridge to connect them, this learning began to seem disjointed. If it's not a computer, it can't run a program, so what are they learning from playing with it? The answer took me back to the research about manipulatives and those early-learning theories I introduced at the start of the article. Learners needed to have a concrete, conceptual understanding of what a computer is before they could start comprehending the more abstract role of a program in that system. We needed to spend more time teaching computing systems.

WHAT IS A COMPUTER? THIS QUESTION HIGHLIGHTS A WEALTH OF ALTERNATE IDEAS ABOUT COMPUTING SYSTEMS

What does that look like?

Even the youngest learners can start learning about what a computer is and how to recognise one. They start with spotting buttons, wires, and batteries, and then talk about what they do. If they recognise that when a button is pressed, there are instructions to follow, they're beginning to understand what a computer is and where you're likely to find them. As children move through lower primary, we can begin spotting buttons and discussing what might happen if we press them. This is where we can start differentiating between things that use electricity and those that run a program.

By upper primary, we explore the world around us and try to work out what the algorithm would be. We use input–process–output to decide if something has a computer inside it (see the next article for more on this). Each time we use this model, we reaffirm what an input and output are, as well as the basic concept of programs running on computers. For example, what's the input on an iPad? How do we tell it what to do? There's a home button and a touchscreen, or we can talk to it using Siri. What code runs when we press the home button? Something like 'when button pressed, show home screen'. And then the output? We can see it on the touchscreen. This simple model allows us to test different machines or items of technology and tell if they're computers or not.

One misconception I regularly hear is children referring to a monitor as a computer. Using this model, we can test this alternate conception. What's the input? There are buttons. What happens when we press them? It says 'no input'. What's the program it's running? It's not doing anything, because there's no laptop plugged in. Then is it a computer? No. We now have a way to start conversations about whether a device is a computer and therefore whether a device is running a program.

Having developed this solution to my problem with teaching computing systems prior to programming, I repeated the 'What is a computer?' question a year later with learners of the same age. This time I got much more varied and detailed responses. Here are some examples:

- A computer has lots of switches and plugs to plug things into; it doesn't have to have a screen
- A computer needs code on a microchip to make it work; without that, pressing a letter would make nothing happen
- Not all computers look like a computer; they have different shapes and designs and are used for different things

While these answers are not perfect, in just a year I was seeing noticeable progress in the complexity of the answers given.

I found similar benefits when teaching programming, where learners could tell me that a wide range of devices ran programs, including Bee-Bots and beyond! Since these early discoveries, I ensure that each September, I start teaching with an age-appropriate introduction to computing systems and make regular links back to this learning when I teach programming later on. The Bee-Bot discussed here was one example of a manipulative, but there are many more examples, from floor robots, to Raspberry Pis, to microcontrollers. There are many ways for you to challenge learners' concepts of what a computer is, including embedded systems where you can find computers in washing machines, traffic lights, or automatic doors.

Learners must learn the ubiquitousness of programming to grow their understanding of a world they're a part of. And as a teacher, once you start these conversations, you never know where they'll end up! Take some time this week to ask your class 'What is a computer?' and carve out time in your curriculum to ensure learners have a foundational understanding of computer systems. (HW)

FURTHER READING

- ✓ Piaget, J. (1952). The origins of intelligence in children (Vol. 8, No. 5, 18-1952). New York: International Universities Press. **helloworld.cc/piaget1952**
- ✓ Bruner, J. S. (1964). The course of cognitive growth. *American Psychologist. 19*(1), 1. **helloworld.cc/bruner1964**
- ✓ Sapounidis, T., & Demetriadis, S. (2013). Tangible versus graphical user interfaces for robot programming: exploring cross-age children's preferences. *Personal and Ubiquitous Computing. 17*(8), 1775-1786. **helloworld.cc/sapounidis2013**

HOW DOES THIS WORK?

The input–process–output model can support young learners with understanding the technology around them

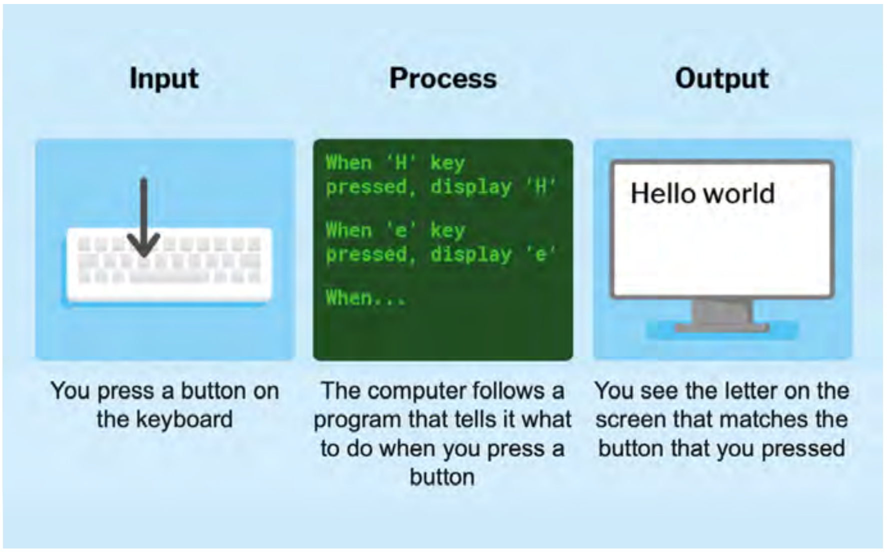
Sometimes, when you ask a question of young learners, you are pretty sure you know what their response will be. Other times, they take you by surprise: you blink, try to keep calm, and begin unpicking exactly where that answer came from. This is what happened when Sway asked a class of seven- to eight-year-olds how automatic doors worked. One child rationally explained that a person was watching the door on a camera, and when someone approached the door, they pressed a button and the doors opened. Simple.

While this is a perfectly reasonable explanation, it is not the correct one. Instead, it highlights that our learners often have no idea how information technology (IT) actually works, what it does, or why something is happening. This makes it seem magical at best, and sentient at worst. This article will explore some common alternate conceptions that learners hold about computing systems, and how you can use the input–process–output (IPO) model to support learners in making sense of the IT around them.

The IPO model

All computers work with inputs, processes, and outputs (see **Figure 1**). All computers accept inputs, which are entered into or received by a computer. They can be generated in many ways, including by a user pressing a key on a keyboard, or a computer receiving a signal from another device. The process then determines what the computer does with that input. It can process the same input in different ways, depending on the program running. The output is how the computer finally presents the results of the process. It can return the results to the user in many ways, such as displaying text on a screen, creating printed materials, or playing a sound from a speaker.

In today's connected world, it's easy to overlook the processes taking place in devices that learners don't immediately recognise as computer systems, such as pedestrian crossings or washing machines. This can lead to learners developing alternate conceptions about what is happening, making it harder for them to apply their understanding of programming or input and output devices as they gain more knowledge. We can't build knowledge on insecure foundations, so the sooner we identify these misconceptions, the better.


Input
Process
Output
Hello world
You press a button on the keyboard
The computer follows a program that tells it what to do when you press a button
You see the letter on the screen that matches the button that you pressed

```
When 'H' key
pressed, display 'H'
When 'e' key
pressed, display 'e'
When...
```

Figure 1 A simple example of the IPO model

Does a lamp have a computer inside it?

As we start to pay attention to the world around us, we begin to recognise different groups of objects that have similar properties, such as natural or manufactured, mechanical or electrical. However, as these objects become more complex, it can be hard to tell which groups they belong to. This ambiguity can make learners overgeneralise their understanding of how something works. Taking time to break this down with the IPO model allows learners to reflect on their assumptions.

Let's imagine a desk lamp. Does it have an input? Yes — I press a button to trigger what I want to happen. Does it have an output? Yes — the light turns on. Now comes the important part: is there a process? No — there is no program receiving data that the button has been pressed. Instead, the switch on the lamp creates a circuit for the electricity to flow through, allowing the bulb to light. Therefore, most lights do not have computers inside of them.

Computers are really clever

The feeling we have that computers are magical, before we start to understand how they work, is often reinforced when the device can do something we do not know how to do ourselves. One of the most prevalent and early alternate conceptions that learners hold about computers is that they are 'really clever'.

To address this, let's consider looking for information on a website to answer a question. What is the input? Using the keyboard to type in keywords that tell the computer what I'd like to know. The search engine's computer then processes this data by running a program to find relevant information. What is the output? A website showing a list of other websites on my screen. Do I have the answer to my question? Most often, no. I now have to go to each web page and decide if it has the answers that I need. Taking learners through each step in the model highlights how much of the process is reliant on human interaction to work, and how computers are only as powerful as the humans that use and program them.

IPO takes place on one device

Without understanding how a system works, it can be very easy to make assumptions. One afternoon, the internet went down at my (Josh's) school. My class, however, didn't believe me! Why? Because the interactive whiteboard was still working. These assumptions become more important when we begin considering personal data, what's stored locally on the device you are using, and what's uploaded to the internet. I've found that this is particularly challenging with certain apps on tablets that may also back up online.

To unpick this, it's important to start considering larger and more complex systems, such as ATMs. The input (data from the keypad) and the output (the information displayed on the screen) are clear. However, much of the process is not happening on the computer within the ATM — it's using the internet. The computer in the ATM sends the input data through the internet to the server at the cardholder's bank, to check it's correct. This is the process. Then the server sends back the output data to show the outcome on the screen. The first data processed will check whether the PIN number is accurate, but each instruction after that will begin the process again. Even if learners can't accurately recognise what's happening on a device and what's happening online, having these IPO conversations can support them in thinking about what's happening before they create content and potentially share it online.

From programming, to collecting data from sensors, to recognising technology around us, the IPO model applies to almost all aspects of computing. Starting activities with the question 'How does this work?' can evolve into learners recognising the many and varied IPO systems in the world around them. You can then get creative, letting learners invent imaginary systems to put the IPO model into practice (see **helloworld.cc/tccsystems1**). Initially, the processes will be assumptions, but as learners' experiences grow, these approaches become a chance for them to imagine the computer systems that will change the world. (HW)

" OUR LEARNERS OFTEN HAVE NO IDEA HOW INFORMATION TECHNOLOGY WORKS, WHAT IT DOES, OR WHY SOMETHING IS HAPPENING

FURTHER RESOURCES

The Raspberry Pi Foundation's free online courses:
- *Teach Computing Systems and Networks to 5- to 11-year-olds*: **helloworld.cc/systemscourse**
- *Get Started Teaching Computing in Primary Schools: Preparing to teach 5-11 year olds*: **helloworld.cc/primarycourse**

Free lessons and activities in the Teach Computing Curriculum:
- Connecting computers (ages seven to eight): **helloworld.cc/tccsystems1**
- Systems and searching (ages nine to ten): **helloworld.cc/tccsystems2**

IPO Model overview document:
- **helloworld.cc/IPOsheet**

JOSH CROSSMAN AND SWAY GRANTHAM

Josh is a programme coordinator at the Raspberry Pi Foundation, working across programmes such as the Teach Computing Curriculum and Hello World. Sway is a senior learning manager at the Raspberry Pi Foundation. She leads a team developing computing resources for primary teachers. Josh and Sway are both former primary teachers (**@SwayGrantham**).

FEATURE

AND OR NOT: GETTING IT RIGHT

Michael Jones brings hands-on Boolean logic gates to his upper-secondary classroom

Boolean logic can be interesting and enjoyable, and can be learnt in a practical way: this was the promise I made to my upper-secondary students.

NOT getting it right

Staring at the first year of a new GCSE syllabus a few years ago, I started wondering how to teach it. In particular, I was concerned about some of the drier aspects, such as logic gates.

To quote our exam board, OCR:

LEARNERS SHOULD HAVE STUDIED THE FOLLOWING:

- TRUTH TABLES
- COMBINING BOOLEAN OPERATORS USING AND, OR, AND NOT TO TWO LEVELS
- APPLYING LOGICAL OPERATORS IN APPROPRIATE TRUTH TABLES TO SOLVE PROBLEMS

The key word here is 'studied'. Traditionally, Boolean logic is delivered with: "This is a NOT gate. This is the truth table for the NOT gate. This is the symbolic representation of the NOT gate." Repeat for the other gates. If this works for you and your students, fine. However, for many students it is not fine, and it turns a learning opportunity into a chore.

I realised that most students could understand the theory, but found it hard to relate this to a real computer, full of circuits and electrical signals. I didn't feel that my students were getting the full value out of this topic. With the demise of design and technology departments across UK schools, we may have lost the link between the circuit and the theory.

AND getting it right

What did we do about it? In that first year of the new syllabus, our investigation of logic gates was just that — an investigation. Going back to the very basics of computer circuits, armed with breadboards, resistors, and TTL (transistor–transistor logic) chips, we undertook a journey into building systems that model Boolean logic. This approach incorporated an exploration of basic electronics, and taught me not to assume that my students understand how circuits work. In planning the sessions, I initially made the mistake of assuming an understanding of resistors, anodes, cathodes, and LEDs.

Working with the basic components enabled my students to embed the knowledge that a circuit is essentially the same, regardless of which logic gate they are creating. This allowed us to conduct blind testing of the chips based on the output produced through the pressing of the two input buttons. The process was very much hands-on and unplugged: not a line of Python or Java in sight, and yet we were programming. If you can encourage students to ask the basic question 'Why does the light go on if I hold down both buttons, but not if only one button is pressed?', it is only a short hop from there to creating the truth table.

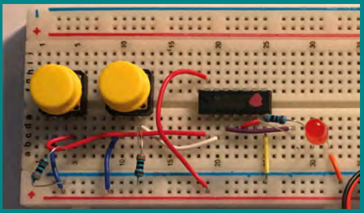
Advanced logic

Early in the process, I recognised that we had an opportunity to go beyond the confines of the syllabus and launch more advanced logic gate investigations, through the development of half and full adders. Using physical circuits that were a natural development of the two-button, one-TTL chip circuit (requiring an additional input chip and some basic components to service the LEDs), we created systems that could add two bits and output the result.

As a result of their explorations, the students now understand Boolean logic. More importantly, they also understand the integral part it plays in computer systems as diverse as calculators and aircraft control computers. (HW)

TTL LOGIC CIRCUIT

With this combination of parts, your students can build and test simple two-input Boolean systems. Just swap out the TTL to change the type of logic gate. Apart from the TTL chips, the components you need are likely to be available from your design and technology department cupboard. Add another chip to test the Boolean logic on two chips/gates in series.



MICHAEL JONES

Michael is the director of computer science at Northfleet Technology College in the UK. He is a CAS Master Teacher, Raspberry Pi Certified Educator, Chartered Information Technology Professional, PGCE Subject Leader, MIT App Inventor Educator, and Specialist Leader of Education (**@MikeJonesCSTalk**).

EXPLORING LOGIC GATES IN MINECRAFT

Martin O'Hanlon shares how you can introduce logic gates to students by creating circuits in Minecraft

Logic gates can be difficult to visualise, and it can be tricky to understand why they are useful in the real world. They are often described as black boxes; their operations being entirely abstract, with the output being the only indication of their function. Minecraft (minecraft.net) is a brilliant tool that can help bring logic gates to life for you and your learners, providing a sandbox to create logic gates from simple components, understand their operation, and connect them to outputs in a virtual world before exploring how they react to inputs.

Building circuits in Minecraft

Before looking at logic gates, it's important to first understand how you can create circuits and transmit binary signals in Minecraft. In the real world, circuits are created by using wires to conduct electricity, and the presence of an electrical voltage (meaning there is electricity flowing through the circuit) indicates that the output is 1, or on. In Minecraft, circuits are created using 'redstone dust', a material used to conduct power (see **Figure 1**), and as in the real world, electricity flowing through the circuit indicates that the output is 1, or on. Input devices in Minecraft (for example, buttons, switches, and pressure plates) generate power that can be transferred to output devices (for example, lamps, doors, and dispensers). For instance, connect a switch to a lamp block using redstone dust and turn the switch on; power is conducted through the redstone dust and the lamp turns on (**Figure 1**).

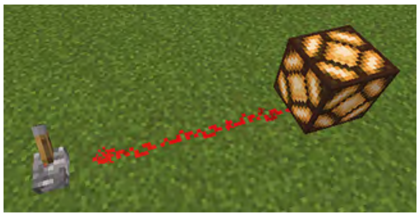


Figure 1 The power from the switch transfers through the redstone dust and turns the lamp on

In addition to redstone dust and input and output devices, there are also components for making more sophisticated circuits, such as redstone torches, repeaters, and comparators. Redstone torches are essential for creating logic gates, as they are not only a source of power, similar to a battery,

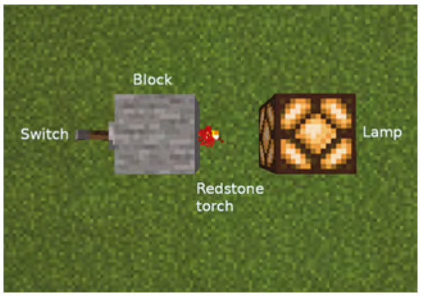


Figure 2 A switch is mounted on a block and connected to a lamp via a redstone torch; when the switch is off, the lamp is on, and vice versa

but they can also invert a signal (for example, if the power is on, the torch will be off, and vice versa). This ability to act like a switch allows you to use redstone torches in similar ways to transistors, the basis of all logic circuits.

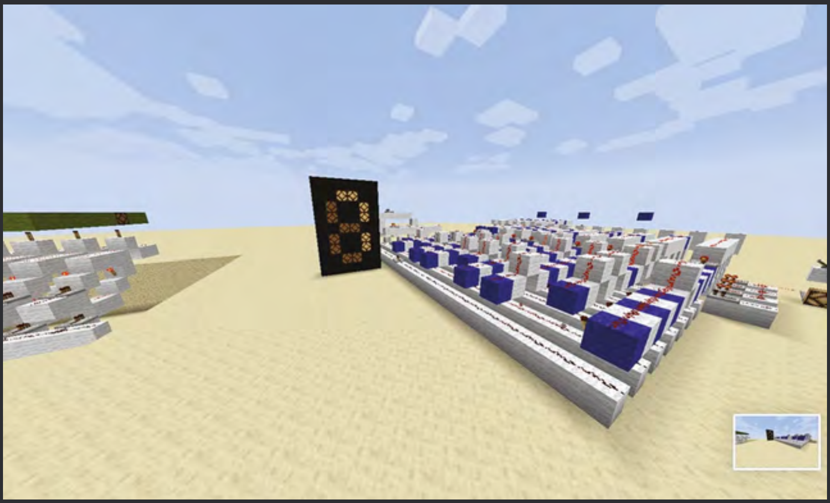
Constructing logic circuits

By connecting redstone dust and redstone torches together in the right configuration, you can create logic gates. The simplest logic gate is a NOT gate whose output is the opposite of the input (for example, if the input is on, the output is off). This can be produced using a redstone torch to invert the input (**Figure 2**).

A REDSTONE COMPUTER

Fully functional computers have been created using redstone in Minecraft by building on top of logic circuits like those introduced in this article.

There are some very impressive builds, including this quad-core computer with user interfaces and displays (**helloworld.cc/minecraftcomputer**).



A practical example of the use of a NOT gate is to create a night light. You can do this by connecting a daylight detector (which is powered when the sun is up) to a lamp via a NOT gate (**Figure 3**). At night, the daylight detector will be off; the NOT gate will invert this signal and turn the lamp on.

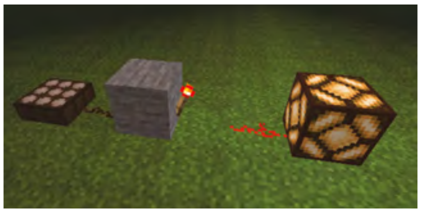


Figure 3 A night light which uses a NOT gate to turn on a lamp when it gets dark

You can also create OR, AND, and XOR gates using redstone torches, dust, and blocks (**Figure 4**):

- An OR gate is simply the connection between two inputs. When either or both inputs are on, the output is also on.
- An AND gate's output is on when both inputs are on. To create an AND gate, you need to use two redstone torches to invert the input connected to a third redstone torch, which will output on when both inputs are also on.
- An XOR (exclusive OR) gate's output is on when either input is on, but off when both inputs are on. The layout of the XOR gate with redstone is more complicated, needing seven redstone torches connected together. This mirrors the complexity of constructing an XOR from NOT, OR, and AND gates.

Investigating how these gates work, and seeing the interaction between the input switches and the lamp, can help learners develop an understanding of each

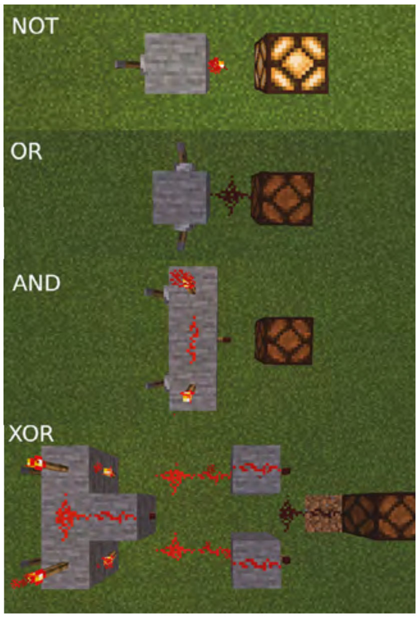
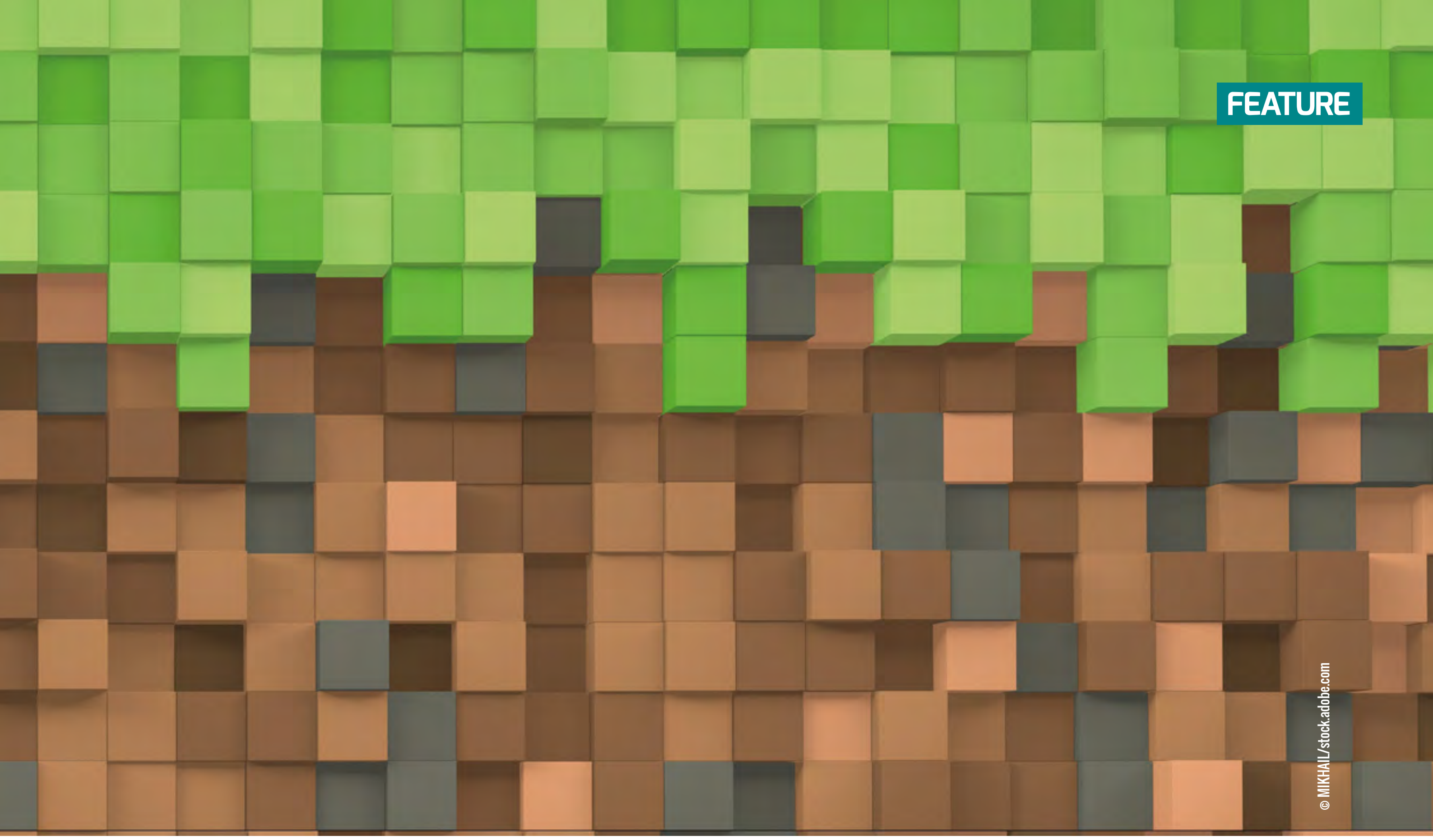


Figure 4 NOT, OR, AND, and XOR logic gates created in Minecraft


© MIKHAIL/stock.adobe.com

gate's operation and act as an engaging introduction to truth tables.

Connecting the Minecraft world

Using these logic gates and different input and output devices in Minecraft, you can create realistic devices, for example, a door that can only be opened by two players at the same time by using two buttons connected to a door via an AND gate (**Figure 5**). This is also an excellent opportunity to introduce logic circuit diagrams, connecting the abstract to the concrete (see the diagram in the top left of **Figure 5**).

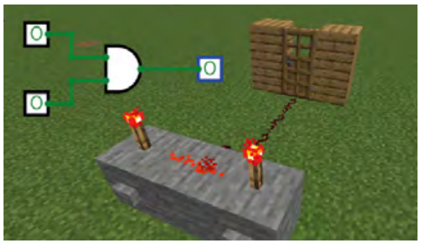


Figure 5 An AND gate is used to create a door that can only be opened by two players working together

Multiple logic gates and circuits can also be linked together to create more complex devices. Take some time to look through the different input and output blocks and components in Minecraft, and let your imagination guide what you create. You may also find it useful to create a logic circuit diagram before embarking on your build.

> REVIEW THE INPUT AND OUTPUT BLOCKS AND COMPONENTS, AND LET YOUR IMAGINATION GUIDE WHAT YOU CREATE

While writing this article, I experimented myself, and created an alarm that beeps when one of my sheep in the Minecraft world leaves its pen. The alarm is automatically activated at night, but also has a switch so I can also turn it on manually during the day (**Figure 6**).

Minecraft and its redstone give you the opportunity to play and experiment with logic gates and circuits, without being constrained by the physicality of creating actual electrical circuits. I hope you have as much fun as I did! (HW)

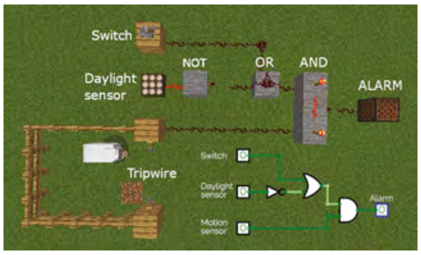


Figure 6 You can create more complex circuits, such as this automatic sheep alarm, by linking multiple gates together

MARTIN O'HANLON
Martin loves technology and creates online learning experiences for the Raspberry Pi Foundation. As a child he wanted to either be a computer scientist, an astronaut, or a snowboard instructor. You will find him on Twitter talking about all these things (**@martinohanlon**).

QUANTUM COMPUTING: AS EASY AS A PENNY FLIP

Andreas J. C. Woitzik and **Stefan Seegerer** introduce the world of quantum information through a quantum penny flip game

Quantum computing has become one of the hot topics in new technologies over the last few years.

Put simply, a quantum computer is "a machine that harnesses some of the unique properties of quantum physics to solve problems that are too complex for regular computers and even supercomputers" (**helloworld.cc/quantumc**). There are strong promises of a possible quantum speed-up of computers, though there are still many obstacles to overcome. In 2019, researchers claimed they had achieved quantum supremacy, a situation in which a quantum computer can solve a specific task much faster than any classical computer could (**helloworld.cc/arute2019**). The specific task they solved was highly artificial and not useful at all, but it showed us that quantum technologies are progressing fast. With many players from industry investing a lot of money and time in this technology, it cannot hurt for students to learn a little more about the topic.

So let us start you off. We are all used to traditional computers with their bits and bytes. Their power stems from being able to very quickly and efficiently manipulate those bits, which we interpret as either 0 or 1. Quantum computing is different: the fundamental bit of information is a qubit (quantum bit). Qubits are used to store quantum information. A qubit can take the values 0 or 1, but more than that, it can also store a superposition of them both. Being in a superposition means that the qubit can be partially regarded as a 0 and partially as a 1 at the same time.

A quantum coin can be in a state of heads, tails, or a superposition of them both

Quantum penny flip game

Students can find quantum computing hard to grasp. To make it more approachable, we can describe superpositions, one of the core concepts, with a quantum penny flip game. David Meyer originally proposed this game in a 1998 paper, but we have created an online version of the game at **helloworld.cc/quantumpenny**. Before you continue reading, give the game a go — it is intended to be played by two people on a mobile phone.

In the game, two people, let's say Alice and Bob, compete in predictions about the outcome of a penny flip. In the classical world, if we spin a coin, there is a 50 percent chance of it landing on either side, heads (0) or tails (1). When it is spinning, we can consider the coin to be in a superposition of heads and tails. In this version of the game, Alice would prepare the penny as either heads or tails, put it in a box, and give the box to Bob. He would then decide whether to flip the box, and return it to Alice. Alice could also choose to flip the box again before opening it with Bob. If the coin showed heads, Alice would win.

In the quantum version, Alice gets a special quantum coin which she can prepare as either heads, tails, or in a superposition of both. The superposition is visualised by a spinning coin. If Alice decides to prepare the coin in a superposition, Bob's flip of the box does not alter the state of the coin. Since Alice can also revert the superposition and Bob cannot change it, she now has full knowledge of the state of the coin and therefore can always win the game.

Further discussions

This game functions as a motivating entry point to the world of quantum computing. While the classical coin behaves like a bit, the quantum coin behaves like a qubit. A subsequent discussion in class allows students to transfer the principle

Students can explore the game, searching for a reliable strategy

to computational ideas. For example, it is easy to see that a bit can be represented by a qubit, as the quantum coin has the states of heads, tails, and the superposition of heads and tails. With that in mind, it can be argued that a quantum computer can perform every computation a classical computer can do, as long as they both have the same number of bits and qubits. But by using its additional states, a quantum computer can, in theory, perform some tasks better than its classical counterpart.

You can also move to discussions of a more advanced algorithm, Grover's algorithm, which uses the concept of superposition to search in an unsorted list. Imagine your unsorted wardrobe contains 100 coats in 100 compartments and you want to find your yellow coat. In a worst-case scenario, you might need to look in all the compartments before finding the coat. Using superposition, though, Grover's algorithm only has to look at the wardrobe ten times before revealing the right position of the object you are looking for (see **helloworld.cc/groversalgorithm** for more). This is useful in a lot of problems that involve an unstructured search.

The penny flip activity can also lead to a discussion about the fragility of quantum information and from there, to quantum cryptography. Once a superposition state is observed, it collapses, and from there, the coin can show only either heads or tails. This feature, an obstacle for those wanting to build a quantum computer, forms the basis of many quantum cryptography protocols. The BB84 key exchange protocol, for example, uses superposition not only to exchange cryptographic keys securely, but also to uncover whether there is an eavesdropper (see page 34 of issue 18 of Hello World for more).

Quantum computing is still very new, and the various unknowns and the abstract nature of it can make it hard for students to grasp. An activity such as the penny flip game, though, can provide an accessible entry point to a technology that may eventually change the way we understand the world. What better incentive could there be to give it a go? (HW)

> “ A QUANTUM BIT, OR QUBIT, CAN TAKE THE VALUES 0 OR 1, BUT CAN ALSO STORE A SUPERPOSITION OF THEM BOTH

ANDREAS J. C. WOITZIK

Andreas is a PhD student in quantum information science in the Quantum Optics and Statistics group at the University of Freiburg, Germany. He is interested in bringing quantum information to schools.

STEFAN SEEGERER

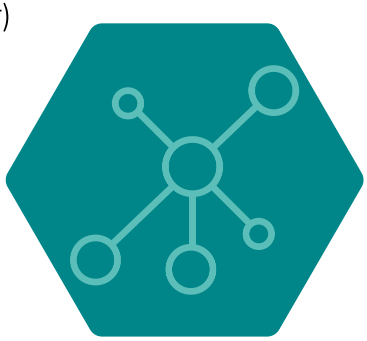
Stefan is the quantum education manager at IQM Quantum Computers, exploring ways to make quantum computing accessible to everyone (**@StefanSeegerer**).

NETWORKS

Networks is a strand very closely related to, and often taught in conjunction with, computer systems. While learning about computer systems is about understanding how these individual systems work, studying networks is about understanding how they work together. In our interconnected world, understanding how networks work is crucial for all computing students.

Many learners will develop experience of using networks before they explore how they work, which gives them an understanding of their value and some of their applications. Learners are likely to begin thinking in broad terms about what a network is and some common components, before exploring the role of addresses, protocols, and so on. Later, they will explore how different networks are physically connected, and how data is transmitted through different layers and across a network.

One challenge this topic presents is that it is very theoretical in nature, as hands-on activities can be impractical in school. However, educators can make the area more concrete by using network simulation tools (such as Packet Tracer) and relating to learners' direct experiences of being connected. This strand is also rich with opportunities for exploring other related strands, including safety and security, and data and information.



IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** networks, in summary
- **What the research says:** the themes and tiers model
- Livening up your networking lessons
- The history of networking
- Using Packet Tracer

IN SUMMARY

NETWORKS

Understand how networks can be used to retrieve and share information and enable global communication

STAGE 1	STAGE 2
■ Describe some uses of the World Wide Web	■ Explain how the internet operates as a global network of networks
■ **Make use of online tools for searching, authoring, and communicating**	■ **Identify internet services other than the World Wide Web**
■ Explain that computers can work together	■ Describe how search engines find, select, and rank results
	■ **Explain that all networked devices have their own IP address**
	■ Explain the need for common methods (protocols) of communication
	■ **Describe how data is transmitted in small chunks or 'packets'**
	■ Identify practical uses of networks along with common network components

In the table below, you will find learning outcomes associated with the ‘Networks’ strand of the Raspberry Pi Foundation’s computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3

- Explain how bandwidth can be used as a measure of network performance
- **Describe and give examples of networking protocols used to provide different services**
- Describe some of the technical vulnerabilities associated with computer networks
- **Explain the role of a firewall in protecting a network**
- Outline the journey of a message sent across a network
- **Identify scenarios where wired or wireless networks are more suitable**

STAGE 4

- Describe the use and structure of different types of network (WAN, LAN, PAN)
- **Explain how physical networks can host multiple associated logical networks**
- Make use of simulations to explore the transmission of data through a network
- **Describe how DNS servers translate URLs into IP addresses**
- Outline how wired and wireless networks can be configured for performance and security
- **Describe how a network can be represented as layers using the TCP/IP model**
- Explain how data passes through each network layer as it is transmitted and received

STAGE 5

- Explain how data transmission is measured and categorised and the factors that affect performance
- **Explain the domain name hierarchy and how domain name servers resolve IP addresses**
- Compare client/server with peer-to-peer communications systems and understand how client/server systems communicate
- **Describe the purpose and function of a comprehensive range of network hardware**
- Describe each layer within the TCP/IP and OSI models and how data passes between them during transmission
- **Describe a wide range of protocols and standards used across a layered network model**
- Explain the relationship between MAC and IP addresses
- **Explain how data is encapsulated at each layer as segments, packets, and frames**

APPROACHES TO TEACHING NETWORKING

STORY BY Katharine Childs, James Robinson, and Andy Bush

We can see the use and application of computing devices in many areas of our world, from the personal devices we use on a daily basis to the systems that control and automate industrial processes. Through these devices, we are almost always connected to a wider network in some form or another. An understanding of how our networks, systems, and devices work is therefore foundational knowledge for all students of computing.

Teach Computing and the Raspberry Pi Foundation propose that we can organise the knowledge and skills within the teaching and learning of networks and computer systems into four broad themes (**helloworld.cc/rpf2021**). In turn, we can then divide these themes into four tiers of detail/abstraction, and the relevant concepts can be mapped to each of the resulting 16 areas as a themes and tiers model (**Figure 1**).

The themes and tiers model

The four key themes are as follows:

HARDWARE

This covers the physical devices and components that work together to form a computer system. The deeper learners explore this theme, the more they focus on how different components work, as well as the logical concepts and physical processes on which the system is built.

SOFTWARE

This encompasses internet services (including cloud computing), operating systems, applications, utilities (such as drivers), and assembly/machine-code language. Learners begin by understanding software that is visible to them, such as the operating system on their computer, before moving onto more abstract software concepts.

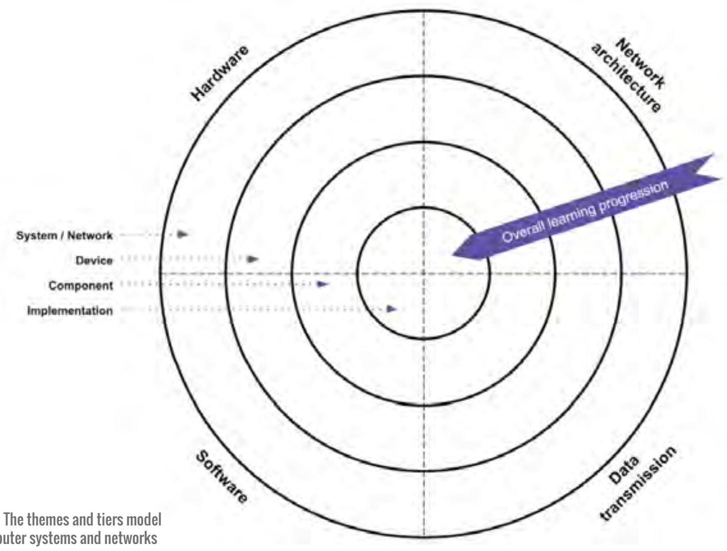
NETWORK ARCHITECTURE

This includes an understanding of the different types of computer network, the components that make up a network, and how these components are connected together. As learning in this theme progresses, learners can design and build their own simple network to a given specification.

DATA TRANSMISSION

This theme focuses on how data moves around networks. Students begin by learning about common protocols for transferring data across networks, and then move on to understanding methods to make sure data is transmitted securely, reliably, and rapidly.

While these themes help describe the content within the study of networks and computer systems, they are still very broad concepts. At different points in a learner's journey, they may explore the same or similar concepts, albeit from a different perspective or level of abstraction. For example, within the data transmission theme, learners may first find out that devices within a network can communicate with each other. Later, they explore the reasons why protocols are needed, and after they delve deeper, they



■ **Figure 1** The themes and tiers model of computer systems and networks

	COMPUTER SYSTEMS		NETWORKS	
	HARDWARE	SOFTWARE	NETWORK ARCHITECTURE	DATA TRANSMISSION
SYSTEM/ NETWORK	Purposes of systems **Benefits of computer systems** Monitoring and controlling systems **Remote storage**	Web services **Cloud computing** Software as a service **Control systems software** Virtual machines	What a network is **Purpose, uses, and risks** The internet **WAN, LAN, and PAN** Network topologies	How and why we use networks **Data can be routed across a network** Network performance, bandwidth, and latency
DEVICE	General purpose vs purpose-built embedded devices **Common device features** Peripherals	Operating systems **Application software** Open/closed source **Human-computing interaction** User interfaces	Devices within the network **Client-server and peer-to-peer networking** Thin and thick clients **Portable devices**	Connectivity **Role of protocols** Application layer protocols **DNS and IP addressing** Servers, email, web, etc.
COMPONENT	Role of common components: • Storage • CPU • RAM, ROM, and cache • Sensors	Utility software **Controlling IO devices** Hierarchy of programming languages	Wired and wireless connections **Switches, access points, routers, gateways, bridges, etc.** NICs and WNICs	Transport layer protocols **DHCP** Network address translation **WebSockets**
IMPLEMENTATION	CPU architecture(s) **Fetch-decode-execute cycle** Logic gates/circuits **Buses** Interrupts **Storage media**	Device drivers and BIOS **Program translators** Assembly and machine-code language **Libraries, linkers, and loaders**	MAC addressing **Characteristics of transmission media, including copper, fibre optic, radio waves, etc.**	Network layered model(s) **Collision detection and avoidance** Multiplexing **Circuit and packet switching**

Figure 2 Networks and computer systems content organised by themes and tiers

become aware of a range of protocols and their uses. Eventually, they become familiar with how those protocols are implemented.

Learners therefore examine the system and wider networks from a range of perspectives. Teach Computing and the Raspberry Pi Foundation refer to these perspectives as tiers, with learners moving from the highest, most abstract tier, to the lowest tier, as follows:

System/network tier: a highly abstract view in which learners focus on how systems and networks are used to solve problems.

Device tier: learners are concerned with familiar computing devices, including computers, phones, tablets, and embedded systems.

“ WE NEED A BALANCE OF THEORETICAL CONCEPTS AND PRACTICAL ACTIVITIES

Component tier: learners look inside the device and understand the purpose of common constituent parts that make up every computing device.

Implementation tier: learners focus on the specific details of how the smallest components are built, how they work, and how they are controlled.

We can see how these themes and tiers can interact in **Figure 2**.

Teaching approaches for computer networks

Teaching about networks requires a balance between theoretical concepts and practical activities, to help learners move from the system tier through to the implementation tier. The theories underpinning computer networks are too abstract to be understood without practical examples, but practical activities alone will not provide the deeper understanding of the principles and protocols that underpin a fully functioning network.

TYPE OF APPROACH	SUBTYPE OF APPROACH
VISUALISATION	■ Using network simulators ■ **Using multimedia and animations** ■ Using visual analogies ■ **Using network monitoring tools**
ACTIVE LEARNING	■ Problem-based learning ■ **Playing games**
HANDS-ON LEARNING	■ Practical activities

Figure 3 A classification of teaching approaches for computer networks

We can classify teaching approaches for learners aged five to eighteen into the seven different categories shown in **Figure 3**, which have been adapted from the work of researchers Prvan and Ožegović (**helloworld.cc/prvan2020**). This categorisation offers a range of approaches that you can try out in your classroom setting, supporting you with striking that balance between theory and practice:

Using network simulators: network simulators help learners to design, configure, and compare network topologies in a risk-free virtual environment. The network design can then be tested for performance, bandwidth, and latency, and modified as required. Examples of network simulation tools include Packet Tracer from Cisco (see the article on page 32 for more on this tool).

Using multimedia and animations: using high-quality video content provides learners with a visual overview of network activity. For example, animations can show some of the paths taken during data transmission, and images can illustrate the different sections of a datagram, including the header and footer.

Using visual analogies: teachers can draw on real-world examples to help explain abstract concepts. An analogy such as the way a letter moves through the postal system can be used to compare the way that data is routed through a network.

When using analogies, it is recommended that a semantic wave approach is used, to unpack theoretical concepts using concrete examples and then repack learners' understanding with clear links back to the theory, to avoid misconceptions (see the 'Unplug, Unpack, Repack' section of *The Big Book of Computing Pedagogy* for more on semantic waves — **helloworld.cc/bigbook**).

Using network monitoring tools: network monitoring tools provide visible information about real-world networks to help learners better understand the process of routing data across a network. This information can be used as a diagnostic tool to think critically about errors in data transmission or to better understand the behaviour of packet exchanges between network layers. Use of these tools is dependent on network security settings, but even basic Windows commands such as 'tracert' can provide interesting learning points about the speed and route of data across networks.

Problem-based learning: identifying errors in a network and fixing them is a type of active learning that can provide opportunities for deeper understanding of concepts. Ensure that the network failures are designed to maximise the opportunity for learning, and to offer opportunities for group discussions to identify potential solutions.

Playing games: game-based learning includes creating scenarios involving insecure or faulty networks and challenging learners to work as a team to problem-solve. This can be a highly motivating context for learners, although teachers must also make sure to pre-teach the prior knowledge learners need and to model examples of successful collaboration.

Practical activities: tasks such as setting up a physical network with low-cost equipment provide valuable hands-on experience to illustrate theoretical principles. For example, we can configure Raspberry Pi to act as a server, and in doing so, learners must set up a static IP address and connect to a default gateway and DNS server. By configuring and testing these settings, learners gain a deeper understanding of how networks are implemented in real-world situations.

There are a number of different practical considerations when choosing a teaching approach for each topic. It is recommended that teachers work with their school IT staff to identify opportunities and constraints, including available equipment, security protocols that are in place on the school's network, and whether high-quality visualisation resources such as videos or animations are available. We should use carefully planned units of work to involve coverage across the themes and tiers model, and to strike the balance between theory and practice, supporting your learners with developing and understanding these foundational skills and knowledge. (HW)

FURTHER READING

- ✓ Teach Computing and the Raspberry Pi Foundation. (2021). *Computer Systems and Networking Within the Computing Curriculum* [White paper]. **helloworld.cc/rpf2021**
- ✓ Prvan, M., & Ožegović, J. (2020). Methods in Teaching Computer Networks: A Literature Review. *ACM Transactions on Computing Education, 20*(3), 1-35. **helloworld.cc/prvan2020**

MAKE NETWORKS INTERESTING WITH FILIUS

Paul Powell shares how you can liven up your lessons on networking

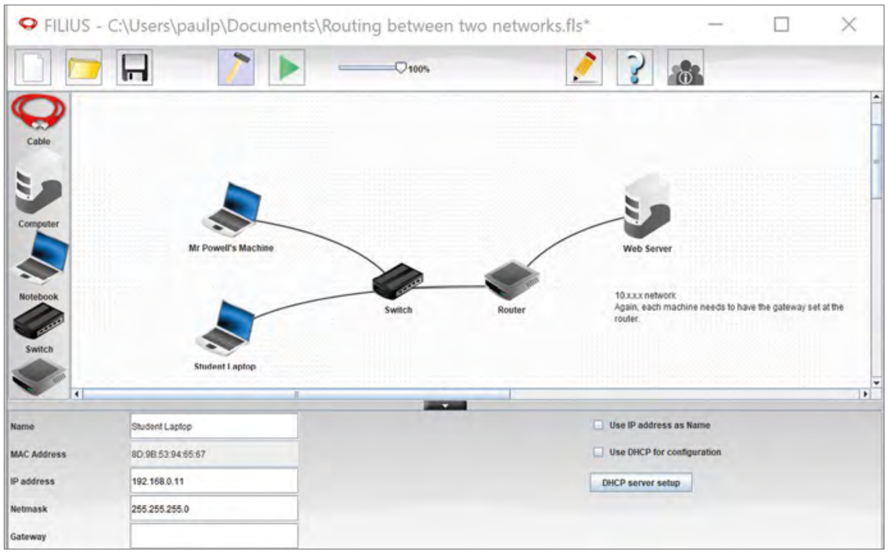
Networking can be dull. Lots of terminology, acronyms, technical detail, and bits of binary. Apart from the underlying sense that it makes YouTube work, what exactly is the point of knowing all this stuff? Some schools are lucky enough to be able to make a network of Raspberry Pis and get them talking to each other, but not every department will have the funds, the space, or the wide range of skills needed.

Filius to the rescue

Trying to get around this issue for my upper-secondary class (aged 15–16), I started looking for a network simulator. Everything I found was either too complex (GNS3 or Packet Tracer) or too restrictive (the Teach-ICT simulator). Eventually I stumbled upon Filius, a good midpoint between the two. Filius is an open-source Java app that was written first in German, and then translated into English, including a decent English guide on the website (**helloworld.cc/filius**).

Filius lets you set up a network with computers, switches, routers, and cables. At the most basic level, this can be used to connect the basic components. Each machine can be set up with an IP address, and then the simulation can be run. When in simulation mode, you can install software onto the machines. You can use this to ping between machines, and you can see the packets going back and forth as green pulses along the wires, or by right-clicking any computer or router and inspecting the packets as they go back and forth.

Once the basics of a LAN are out of the way, Filius then lets you set up multiple networks, routing tables, web servers and web browsers, email, DHCP, and more. This might all sound a little daunting, especially with classes that are a little less than attentive. Fortunately, Filius lets you load and save your networks, so you can prepare them in advance to teach specific concepts.



10.x.x.x network
Again, each machine needs to have the gateway set at the router.

Name	Student Laptop
MAC Address	8D:9B:53:94:65:67
IP address	192.168.0.11
Netmask	255.255.255.0
Gateway	

Two connected networks showing client/server communications

Making it accessible

Adapting the concept of the PRIMM (Predict–Run–Investigate–Modify–Make) approach to programming, I tried to structure the activities so that students began with a supported activity. Typically, this meant giving students a network that was already set up, with an element of the network working. Students had to predict what the network would do (mostly based on machine names and topology) and simulate (run) the network, as well as carrying out a few tasks, investigating the settings, modifying the settings of a non-working portion to get it running, and then making a new section of the network.

This approach was very successful with my class. Everyone was engaged, and it helped explain the different forms of addressing. The next step will be to break the tasks down further so that any students who are struggling can begin to work more independently. (HW)

PAUL POWELL

Paul currently works as a solution delivery manager for a software development provider. Before this he was a computing teacher in the UK for ten years.

THE JOURNEY OF NETWORKING

To help our young learners understand the internet better, we need to follow the development of networking from the first connected computers

The power of a single computer is well understood. The speed and accuracy with which it can run through an algorithm or solve the most complex problems is outstanding. However, it's only the tip of the iceberg when compared to the power of networked computers.

The internet is the biggest worldwide network of devices, and has truly transformed almost every aspect of how we live, work, and play. Virtually all our infrastructure is reliant on this global connectivity. The ubiquitous nature of its availability and ease of access has resulted in it replacing costly dedicated connections and transforming just about everything. Electrical generation, water distribution, transport networks, banks, government, the media, and most of education would cease to function with no network connectivity. Children might relate to the idea of losing Facebook or Snapchat, but fail to realise there would be no power distribution and little food in the shops if there was no network.

The networks we have today and the rules they follow are partially a legacy of older technology. If we had a clean slate and could start again, we wouldn't build the internet we have today. Hence, to understand the internet, we need to follow the development of networking from the very first connected computers.

Early connections

As computers developed, it was recognised that they could be connected to share information. With two computers, it was easy to provide a dedicated link between them and use electrical voltages to represent the binary 1s and 0s of the data to be sent. The electrical signals are referred to as 'layer 1' or the 'physical layer' because they're the closest to the physical connections.

Three computers can be connected with three connections. Four computers need six connections, and so on. The advantage of this system was that each computer could choose where to send the information, just by selecting the appropriate connection. The downside was the number of connections and thus connectors on each device; a fully meshed network (**Figure 1**) of six computers would need five connections from each computer and fifteen connections in total. Imagine what 100 connected computers would look like! What was needed was a way of using a single wire that all devices could connect to, and a way of somehow sharing usage of the wire between them all. Two distinct solutions evolved: the 'ring' and 'bus' topologies.

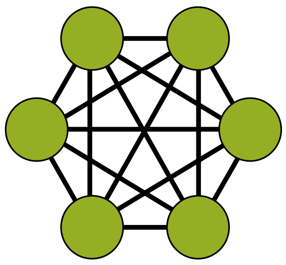


Figure 1 Fully meshed network

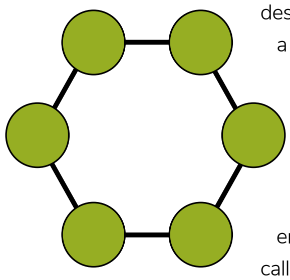


Figure 2 A ring topology

Ring

Rather than connecting each computer directly to every other computer, they can be connected in a ring topology (**Figure 2**). The data passes round the ring, with each computer 'seeing' the data and also passing it on. The computer that puts the data in the ring can remove it if it returns. But how do we ensure the data only goes to the computer we want to send it to, as it now goes through all the computers?

The solution involves giving each computer in our ring a unique address. Before we send our data out, we add the destination address to the front of the data. For the receiving computer to know who to reply to, we also add our own address as the source address.

This addition of a source address and destination address to the data forms a 'frame'. The process of adding this additional data is called 'encapsulation'. It's similar to placing the data in an envelope and putting the destination address on the front and the sender's address on the back. The process that encapsulates the data with addresses is called 'layer 2' or the 'data link' layer. The format of the frame is therefore:

SOURCE ADDRESS	DESTINATION ADDRESS	DATA

As the frame is passed around the ring, each computer compares the destination address to its own address. If there's a match, the computer reads the frame and has received the data. Computers that don't match the address just forward the frame on.

Ring-type networks are used today in synchronous optical networking (SONET) and synchronous digital hierarchy (SDH) networks.

Bus

Another option to connect multiple computers together is to connect them all to a common wire. Initially, this was a thick coaxial cable, similar to a TV cable. Each computer was connected to the cable with a 'tap', which was a spike in a clamp, tightened up with a nut. This visualisation of many computers being connected to a common wire inspired the name 'bus topology', with the idea of a bus that people can get on and off as they wish (**Figure 3**).

The single wire meant that only one computer could send data at any one time, and the data would go to every computer on the wire. The technology was called 'ethernet' and used a set of rules called CSMA/CD to manage what would happen when more than one computer tried to send data at the same time. Mechanical issues with the taps and dry joints led to these networks being unreliable and difficult to fix. An improvement used a thinner coaxial cable and special connectors called BNC connectors. This was called 'thin ethernet', and the original cabling was retrospectively renamed 'thick ethernet'.

Just like in a ring network, all the computers need an address. This is the media access control (MAC), ethernet, physical or hardware address. Data is encapsulated, with a header containing the source and destination address, to make a data frame. In early ethernet networks, the frames were received by all computers, and each compared the destination address to its own address. If there was a match, the computer read the frame and received the data; otherwise, the frame was just ignored. The format of the frame is therefore:

SOURCE ADDRESS (6 bytes)	DESTINATION ADDRESS (6 bytes)	DATA (up to 1500 bytes)

Cabling

With thin and thick ethernet networks, the electrical signals that carried the binary data were protected from interference by the braiding in the coaxial cable. This shielded the inner conductor by providing an electrical Faraday cage around the core.

Improvements in technology meant the data could be sent over a pair of wires twisted together in the same way as wires are twisted in a telephone cable. One pair is used to transmit data, and the other to receive it.

The cable is called an 'unshielded twisted pair' (UTP), and is commonly, although incorrectly, known as an ethernet cable. Connection is made via RJ45 plugs.

RESOURCES TO VISUALISE NETWORKS

DESCRIPTION	VIDEO	SCRATCH ANIMATION
Fully meshed networked computers	helloworld.cc/meshedvideo	helloworld.cc/meshedscratch
Ring-based network	helloworld.cc/ringvideo	helloworld.cc/ringscratch
Ethernet bus-based network	helloworld.cc/busvideo	helloworld.cc/busscratch
Ethernet hub-based network	helloworld.cc/hubvideo	helloworld.cc/hubscratch
Ethernet switch-based network	helloworld.cc/switchvideo	helloworld.cc/switchscratch
NIC sending data frame	helloworld.cc/NIC1video	helloworld.cc/NIC1scratch
NIC receiving data frame	helloworld.cc/NIC2video	helloworld.cc/NIC2scratch

Hub

To provide resilience and simplify connections, the bus was collapsed into a box called a hub. Each device connected directly to the hub on its own RJ45 port. Inside the hub, signals were received on one pair; they were then regenerated, and just like the bus, were transmitted out of all ports. This simple, reliable, and cheap way to connect computers led to a high growth in the number of local area networks (LANs) with multiple computers connected using an ethernet hub. We call hubs 'layer 1' or 'physical layer' devices, because they just regenerate the electrical signals with no notion of the structure of the frame.

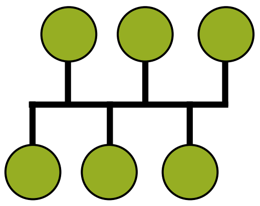


Figure 3 Bus topology

Switch

Hubs just forward data frames out of all ports, because they have no knowledge of which computers are connected to which ports, and have no understanding of the data they're forwarding. However, advances in electronics have allowed us to improve the efficiency of our ethernet networks by putting some intelligence in the hub. They can now inspect the frame and examine the source and destination ethernet addresses. Clearly, the device is now much more than our humble hub, and is called a 'switch'.

Initially, the switch will not know the addresses of the connected computers, so it defaults to hub behaviour and switches incoming frames out of all ports. However, it learns which addresses are connected to which ports by examining source addresses on incoming frames, which are stored in a table within the switch. Hence, future frames are switched to the right ports. We call switches 'layer 2' devices because they understand the headers at layer 2, the data link layer.

The function of encapsulation is provided by the network interface card (NIC) in the computer. Different interfaces, such as wired, wireless, and 3G/4G, will all have different NICs. No matter the media (except fibre), there's still the possibility of some electrical interference with the signal, and spikes in the voltage. These spikes can result in a binary 0 being interpreted as a binary 1, or vice versa. It may not be obvious that an error has occurred, so we use a 'check-field' at the end of the frame to enable us to detect errors. Thus our ethernet frame is now:

SOURCE ADDRESS (6 bytes)	DESTINATION ADDRESS (6 bytes)	DATA (up to 1500 bytes)	CHECK-FIELD

When receiving the frame, we check the check-field to see if any errors have occurred. If an error occurs, we discard the frame.

The next evolution was the interconnection of all these LANs and the birth of the internet protocol (IP) … but that's another article! When trying to explain how the internet works, it's vital to appreciate that it has been a journey over some six or seven decades — and we hope this article has been a helpful start! (HW)

DUNCAN MAIDENS
Duncan is the director of computer science education at the Raspberry Pi Foundation.

THE PRINCIPLES OF NETWORKING MADE EASIER WITH PACKET TRACER

Cisco's powerful simulation tool encourages practice, discovery, and troubleshooting, and lets students experiment with network behaviour by building complex networks – plus it's free to schools!

A reliable network forms the heart of any technology infrastructure, and networking is now at the forefront of technology innovation in our increasingly digital world. Teachers must therefore be equipped to teach networking in classrooms, particularly given the skills shortages in this area. With many teachers struggling to tackle networking principles, purchasing switchers, routers, and other such devices might appear to offer the perfect solution. For most schools, however, this isn't a practical or affordable option, which is why Cisco has created the next best thing for teaching networking — Packet Tracer (helloworld.cc/packettracer).

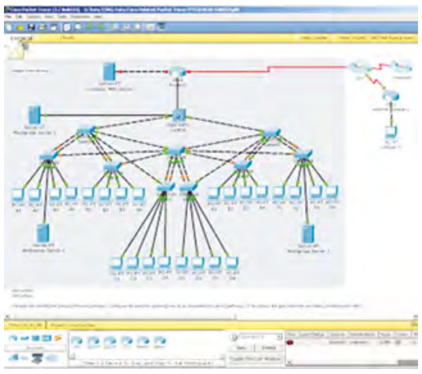
CISCO NETWORKING ACADEMY

Founded in 1997, Cisco Networking Academy (**helloworld.cc/ciscoacademy**) is a not-for-profit IT skills and career building programme that connects millions of students, educators, and employers worldwide. As part of this programme, Cisco partners with learning institutions to deliver technical training and problem-solving experiences to individuals studying networking, security, and IoT technologies.

What is Packet Tracer?

Packet Tracer simplifies the complexity of teaching networking, while giving students valuable hands-on experience. Cisco developed Packet Tracer to help its Networking Academy students achieve the best learning experience while gaining practical networking technology skills.

Packet Tracer is a powerful simulation tool that students can use to build, explore, and troubleshoot a variety of network environments as if the hardware were with them in the room. By dragging and dropping routers, switches, and various other types of network devices, they can develop virtual network worlds. This paves the way for teachers and students to explore, experiment, and discover an almost unlimited array of networking concepts and technologies.


Solving puzzles collaboratively enables student-led learning

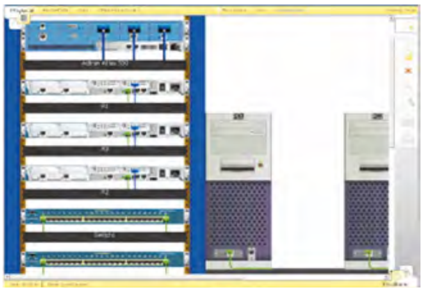
Key features

Packet Tracer has two workspaces — logical and physical — which you can easily switch at the click of a button. The logical workspace allows users to build coherent network topologies by placing, connecting, and clustering virtual network devices. The physical workspace provides a graphical physical dimension of the logical network, giving a sense of scale and placement in how network devices such as routers, switches, and hosts would look in a real environment. The physical view also provides geographic representations of networks, including multiple cities, buildings, and wiring closets.

Packet Tracer also gives you two operating modes to visualise the behaviour of a network: real-time mode and simulation mode. In real-time mode, the network behaves as real devices do, with immediate real-time responses for all network activities. Simulation mode gives students a viable alternative to real equipment and allows them to get configuration practice before working with physical equipment.

Supports teaching networking

The Packet Tracer tool contains many exciting features that offer an extra dimension to teaching computing. Schools can easily teach and demonstrate complex technical models, as well as networking concepts and protocols, using an interactive environment.


The physical workspace offers a graphical view of the logical network

Multiuser games provide fun learning opportunities for collaboration and competition

One key feature, the Activity Wizard, allows teachers to write their own learning activities by setting up different scenarios. They can customise these scenarios with instructional text, while creating initial and final network topologies and predefined packets. The Activity Wizard also includes grading and feedback capabilities. In addition, you can save and share activities with other teachers and students.

Packet Tracer helps the teaching of networking by:

- Providing a visual demonstration of complex technologies and configurations
- Letting teachers author customised, guided activities that provide immediate feedback through the Activity Wizard
- Facilitating numerous learning activities, such as individual and group lab activities, homework, assessments, games, network design, troubleshooting, modelling tasks, case studies, and competitions
- Allowing self-paced learning outside the classroom
- Supporting social learning processes by enabling collaboration and competition

The student experience

Packet Tracer's hands-on approach to learning means students will be better equipped to apply concepts and configuration fundamentals when exposed to real equipment. By experimenting with network behaviour and asking 'what if' questions, students will gain a solid understanding of how devices connect and communicate in a live network, and how data flows from one device to another.

The software uses a drag-and-drop user interface, allowing students to add and remove simulated network devices as they wish, and lets students practise using a command-line interface. This is a fundamental component of learning how to configure routers and switches. Just as the physical equipment allows you to modify hardware, Packet Tracer offers the ability to insert interface cards into modular routers and switches, which then become part of the simulation.

“ BY EXPERIMENTING, STUDENTS WILL UNDERSTAND HOW DEVICES CONNECT AND COMMUNICATE IN A LIVE NETWORK

Students can also learn how to design complex and large networks, which isn't always possible using physical hardware. From the very basics, such as connecting a PC to a hub, or setting up a server and building a local area network (LAN) and wide area network (WAN), students can build an almost unlimited number of environments. And as they gain practical experience of configuration, troubleshooting, and other tasks, they become more confident in their abilities. In addition, the simulation-based learning environment helps students develop essential business skills, such as decision-making, creative and critical thinking, and problem-solving.

To access Packet Tracer and explore a range of networking activities suitable for the classroom, sign up for one of the free courses at **helloworld.cc/packettracer**. If you want to know more about becoming a Cisco Networking Academy, which gives you access to extra learning materials and activities, and professional development such as Python, C++, C, cybersecurity, and Linux courses, visit **helloworld.cc/ciscoacademy**. (HW)

HOW DOES PACKET TRACER HELP YOU TO TEACH NETWORKING?

Packet Tracer allows users to build and configure a functioning computer network in a simulated environment. Traffic is simulated, with web and email servers being used by a variety of desktop applications. Simulations work in either real-time mode, as they would in a physical network environment, or in simulated mode, where users can see the actual packets and frames moving through the network and decode the layers structure of ethernet, IP, and TCP.

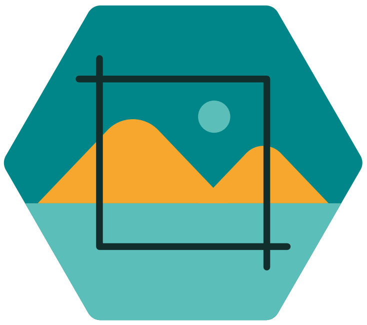
Duncan Maidens, director of computer science at the Raspberry Pi Foundation

HELEN CLOTHIER
Helen is a Country Digital Acceleration skills programme manager at Cisco.

CREATING MEDIA

An important set of digital skills that learners develop through their computing studies is the ability to work with a variety of media, from text, to 2D and 3D graphics, to audio and video, to interactive media. Whether developing new media or integrating and combining existing content, learners encounter a range of common concepts and skills, including grouping, layering, and alignment. Additionally, learners should develop an understanding of how different media are represented and stored by a computing device. This aspect connects to other strands, providing context while also helping learners to evaluate the relative merits of different media formats and consider factors such as compression and file types.

The journey for learners will probably begin with simple and familiar media, including text and images, and gradually expand into a broader mixture of media such as animations, 3D models, and videos. Once they are comfortable with a selection of media types, learners can become more selective about the suitability of different media for different projects. As they progress into the later stages of their computing education, their ability to create media will be valuable, but unless they are taking a specialised media-related qualification, it is unlikely to be the focus of their computing studies.



IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** creating media, in summary
- **What the research says:** threshold concepts in creating media
- Art, creativity, and computer science
- 3D-animated films
- Separating the learning from the tool

IN SUMMARY

CREATING MEDIA

Select and create a range of media including text, images, sounds, and video

STAGE 1

- Identify different forms of media, including text, images, video, and sounds
- **Explain that text can be displayed in different styles**
- Contrast the strengths and drawbacks of using technology to create media
- **Make use of a range of input devices to create and capture media**
- Combine text and images in documents

STAGE 2

- Distinguish between examples of vector/bitmap images
- **Distinguish between an editable multimedia project and the exported media it can produce**
- Make use of layering when working with images, sounds, and video
- **Explain how grouping can be used to work with multiple digital objects**
- Select and apply suitable text formats for a range of purposes
- **Explain what 'good' looks like for a particular digital artefact**
- Capture and edit images, sounds, and video for a given purpose
- **Create multimedia including sounds, images, video, and 3D objects**
- Select, manipulate, and arrange multimedia for a purpose

In the table below, you will find learning outcomes associated with the 'Creating media' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Explain the use cases for both vector and bitmap images	Explain how the technical properties of different media affect a finished product	Analyse digital media products, identifying their purpose and intended audiences
Describe how vector and bitmap images are stored	**Describe physical factors (such as lighting or noise) that affect the quality of recorded or captured media**	**Propose, design, create, and evaluate digital artefacts using a range of media**
Explain the factors that affect the quality of collected or created media	Describe the means by which text, images, and sounds are represented using binary numbers	Describe how both vector and bitmap images are created
Identify ways in which digital artefacts can be manipulated and the motivations for doing so	**Apply a wide range of techniques to compose and enhance digital artefacts**	**Describe the process by which sound (analogue) is sampled and stored digitally**
Decide from examples what makes specific digital artefacts 'good'	Develop and apply templates to speed up production and improve consistency	Calculate the expected file size of different media based upon their attributes
Describe how digital artefacts can be made more usable and accessible	**Find and create digital assets that are suitable and compatible with a final product**	**Describe the impact of changing media attributes (such as colour depth or sample rate) on quality and storage needs**
Combine software tools to create digital artefacts		Choose appropriate media formats comparing quality, file size, and performance
Create or adapt digital artefacts to make them suitable for different audiences		
Apply consistent styles and common assets to give digital artefacts a shared identity		

Understanding threshold concepts is vital if learners are to progress and build subsequent skills and concepts

THRESHOLD CONCEPTS

STORY BY Ben Hall

Even if you have never heard of or used the term 'threshold concepts', you will be teaching them day in, day out, both in your computing classes and in any other subjects you may teach. Researchers Meyer and Land introduced the term to education in 2003, stressing the importance of the idea as "akin to a portal, opening up a new and previously inaccessible way of thinking about something. It represents a transformed way of understanding, or interpreting, or viewing something without which the learner cannot progress" (helloworld.cc/meyer2003). This article will explore the research behind defining and identifying threshold concepts, with a focus on 'Creating media' topics, before suggesting some approaches to introducing them to your classroom.

Identifying threshold concepts

Researcher Peter Davies suggests two methods for identifying threshold concepts (**helloworld.cc/davies2006**). The first approach includes the engagement of two distinct disciplines, and specifically, the views in which these disciplines examine the same situation. The second approach, which is mostly used in the current literature, such as Janet and Nathan Rountree's 2009 paper (**helloworld.cc/rountree2009**), suggests that to identify threshold concepts, the researcher should concentrate on people inside and outside of the community; that is, on the different ways in which students and experts experience the situation.

The Raspberry Pi Foundation has defined threshold concepts within computing education in a similar way. When deciding whether or not a concept is a threshold concept, we at the Foundation use the following criteria:

- A threshold concept should be relevant to two or more topic areas within the subject (portability)
- A threshold concept should be revisited from a teaching and learning perspective several times across different ages and stages of learning (buildability)

In some subjects, threshold concepts are easy to identify. Consider young readers, for example. They are initially taught the threshold concept of phonics, which introduces them to the alphabet and letter sounds, which they can blend into simple and then more complex words. They are then introduced to the threshold concept that not all words are spelled as they sound.

The same is true in computing, or at least some elements of it. In programming, in the Teach Computing Curriculum, we introduce four threshold concepts, one per year group from ages seven to eleven: sequence, repetition, selection, and variables (**helloworld.cc/tcc**). By isolating these four concepts, you can introduce each progressively, and gradually develop an understanding of each one. We have found that there is a broad consensus that this order is appropriate at this age range.

In some areas of computing, however, threshold concepts are not as widely agreed on or accepted. The content and curriculum team at the Raspberry Pi Foundation has picked out 'Creating media' as an example of a strand of computing that could benefit from an approach more based on threshold concepts. At a high level, we have identified a number of commonalities when working with computer-based media. Whether you're creating a presentation or producing a video, you will need some understanding of:

- Files
- Text
- Images
- Audio/video
- Animation
- Layers
- Objects
- Hyperlinks
- Preview
- Templates

There may be more concepts than this; research in this area of computing education is underdeveloped, so we aim to narrow down to the key concepts. Some of the items in the list above can be considered more skills-based than others, for example file management. But there are some clear concepts that can be considered as thresholds to further learning:

- Layers
- Objects
- Hyperlinks
- Preview
- Placeholders
- Templates/styles (global application)
- Colour accessibility

Once identified, these concepts can be introduced progressively across a range of media, teaching students concepts and skills that are transferable across different areas of the computing curriculum. For example, if a student understands layering in desktop publishing, they will be at a significant advantage when they move on to creating vector drawings, as the concept is fundamentally the same in both types of media. This can deliver significant benefits to students across all age ranges:

- The identification of threshold concepts accentuates their importance
- Skills and concepts can be introduced systematically in one context before being applied to others
- The transition between learning stages can be smoother, with less need to recap
- Students learn to apply concepts across different curriculum areas

Pedagogical strategies

Now that we have considered how we define and identify threshold concepts, we need to look at the best way of teaching them to learners and the pedagogical strategies that might be most suitable for threshold concepts in the 'Creating media' strand of computing. In programming, there is a great deal of research into the best ways of teaching new skills and concepts. Thousands of educators have implemented and iterated strategies such as Use–Modify–Create and Predict–Run–Investigate–Modify–Make (PRIMM). In 'Creating media', there is nowhere near as much to go on. So, is there scope to adapt some of the strategies that are so widely used in programming to other areas of the curriculum?

Let's consider the Use–Modify–Create strategy introduced by researcher Lee and her colleagues, which gives students a structure for learning to program (**helloworld.cc/lee2011**). At the Use stage, students use an existing program, analysing what it does and how it does it. They then apply this knowledge to modify parts of the program so that it achieves a different outcome, before creating their own program from scratch.

Could a similar approach work in 'Creating media'? Instead of Use, for example, could students 'consume' a particular type of media? Take vector drawings as an example. In isolation, many students will not be familiar with the concept of vector drawings. To familiarise them with it, you could ask learners to edit or adapt a vector graphic so that it suits a different audience or purpose (that is, as part of the Modify stage). This could involve changing the position, order, or colour of objects, but crucially, not creating anything new.

At this point, you could cover threshold concepts associated with layers, objects, and colours, without the added cognitive load of students having to think about creating their own idea. They could then build upon this experience to create a new artefact, with the scaffolding of existing content. It may be that you are actually already doing some of this in your own practice, but having a structure could help you to formalise and standardise your approach.

It is vital that we know how to define, identify, and approach the teaching of threshold concepts in the computing classroom, to ensure that learners can progress and build subsequent skills and concepts successfully. Evidently, there is still work to be done within this area for 'Creating media' topics. How could you apply these ideas in your classroom, and are there any other ideas you might be able to weave in? (HW)

> 'CREATING MEDIA' TOPICS COULD BENEFIT FROM A THRESHOLD CONCEPT APPROACH

FURTHER READING

- ✓ Meyer, J., & Land, R. (2003). *Threshold concepts and troublesome knowledge*. ETL Project. **helloworld.cc/meyer2003**
- ✓ Davies, P. (2006). Threshold concepts: How can we recognise them? In J. Meyer, & R. Land (Eds.), *Overcoming barriers to student understanding*. Routledge. **helloworld.cc/davies2006**
- ✓ Rountree, J., & Rountree, N. (2009). Issues regarding threshold concepts in computer science. *Conferences in Research and Practice in Information Technology Series, 95*, 139-146. **helloworld.cc/rountree2009**

ART, CREATIVITY, AND COMPUTER SCIENCE

Paul Curzon, **Jane Waite**, and **Ged Gast** argue that art and computing have a lot more in common than you might imagine

Both computing and art involve creativity, innovation, and imagination. And people who possess both artistic and computing skills can make wonderful things happen.

Art involves a creative process, and creating an emotionally or intellectually engaging work of art requires more than just skill with a chisel or paintbrush and an understanding of the medium. It needs innovation, creativity, and imagination. To most people, this may sound a million miles from the stereotype of the computer programmer, whose work is based on logical thinking, but the links are deeper than you might imagine. Computer science also involves great creativity, innovation, and imagination.

On top of these high-level similarities in approach, artists are increasingly using digital media, and this opens new opportunities for interactive art for those with programming and electronics skills.

© NoDenmand/stock.adobe.com

The original iPhone keyboard represented a breakthrough with its ability to have the software step in and help the user

The creative computer scientist

Programming is obviously creative in the loose sense that it involves constructing new things, but it is also creative in a more inventive sense. When programming, you are not creating things by rote, and not following a fixed plan; you are devising something that has never existed before. If you approach opportunities creatively, you may even completely change the way the world does things.

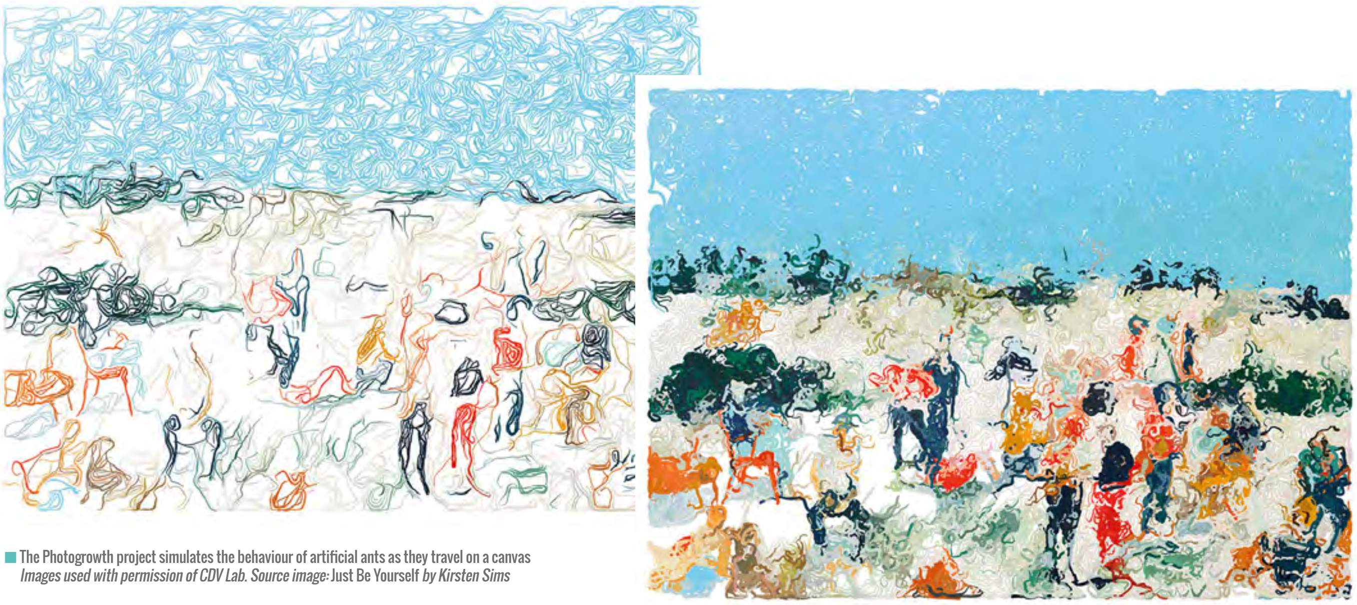
Take the original iPhone keyboard. Before its launch, Apple had a problem: the virtual keyboard was unusable. The buttons were just too small. Lots of engineers worked on the problem, trying out different keyboard layouts, but nothing worked. With the launch looming, the situation looked dire. Without a usable keyboard, the product would flop. Then someone thought about it differently. Rather than using a different layout, they had the idea of writing a program that would predict which letter was most likely to come next, based on those that had gone before, and making the area of those keys larger. It worked, and the iPhone went on to become one of the company's biggest ever successes. It took creativity, innovation, and imagination to come up with this winning solution, and skill with the medium to make it work.

Art and the machine

Artists now have new media to play with. By using programs and electronics as media, they can create interactive art. For example, Soda Constructor was a program that implemented a simple 2D line-drawing program. However, it also added in the laws of physics. Points were masses, and lines were springs with rules based on Newton's and Hooke's laws. Springs could also be turned into muscles that were given periodic energy boosts, making them stretch and contract. Users made a digital drawing and then switched on the laws of physics, and their pictures would come alive. You could even make creatures, following all kinds of body shapes and means of locomotion, that moved around the screen. The result was an amazing program that won a BAFTA for interactive art.

> ARTISTS ARE INCREASINGLY USING DIGITAL MEDIA TO CREATE INTERACTIVE ART

Interactive programs create imagery on a computer screen, but digital art can also escape into the real world. Physical installations can be computer-controlled, creating, for example, sound and light

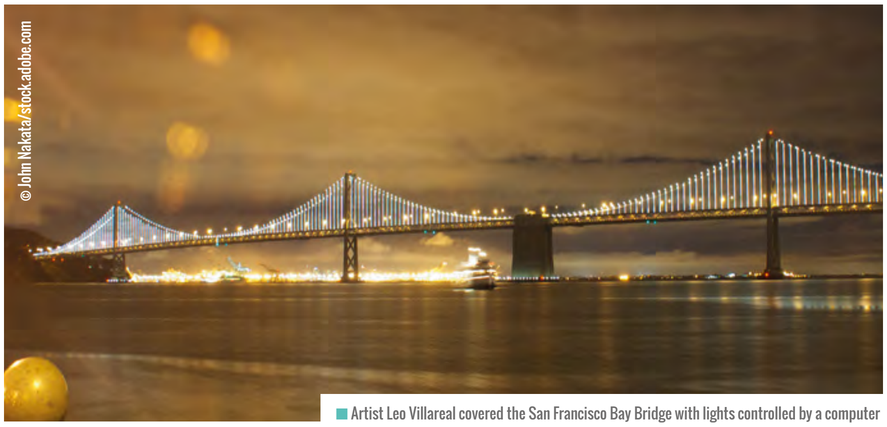


The Photogrowth project simulates the behaviour of artificial ants as they travel on a canvas
Images used with permission of CDV Lab. Source image: Just Be Yourself *by Kirsten Sims*

shows, where sculptures include light and sound effects controlled by sensors. Epic-scale examples include the work of artist Leo Villareal, who turned the San Francisco Bay Bridge into a pulsating, ever-changing work of art by covering it in lights controlled by a computer. Artist Di Mainstone had a different approach to turning bridges into art. At Tower Bridge in London, she put digital sensors on bridge cables attached by lines to a performer's clothing. As the bridge vibrated with traffic and people, and the performer moved, the angle and length of the lines were measured and different sounds produced. Human and bridge thus became one augmented artistic instrument.

Another example of the use of digital creative practice in art is the wonderful sculpture The Hive, at Kew Gardens in London. It is an architectural-scale sculpture in the shape of a walk-in beehive. It is covered in lights that pulse, controlled by the activity of bees in hives in the Gardens.

This kind of art often uses very simple combinations of sensors, lights, and sounds with simple computer control. The artist is writing programs as an essential part of the creative process. As an artistic medium, it is now within the reach of school students, combining programming, electronics, and art. If you have a creative spark, you can make similar kinds of art with simple sensor kits such as Arduinos and Raspberry Pis.



Artist Leo Villareal covered the San Francisco Bay Bridge with lights controlled by a computer

PAUL CURZON

Paul is a professor of computer science at Queen Mary University of London and co-founded both cs4fn (**cs4fn.org**) and Teaching London Computing (**teachinglondoncomputing.org**). He is author of the book *The Power of Computational Thinking*.

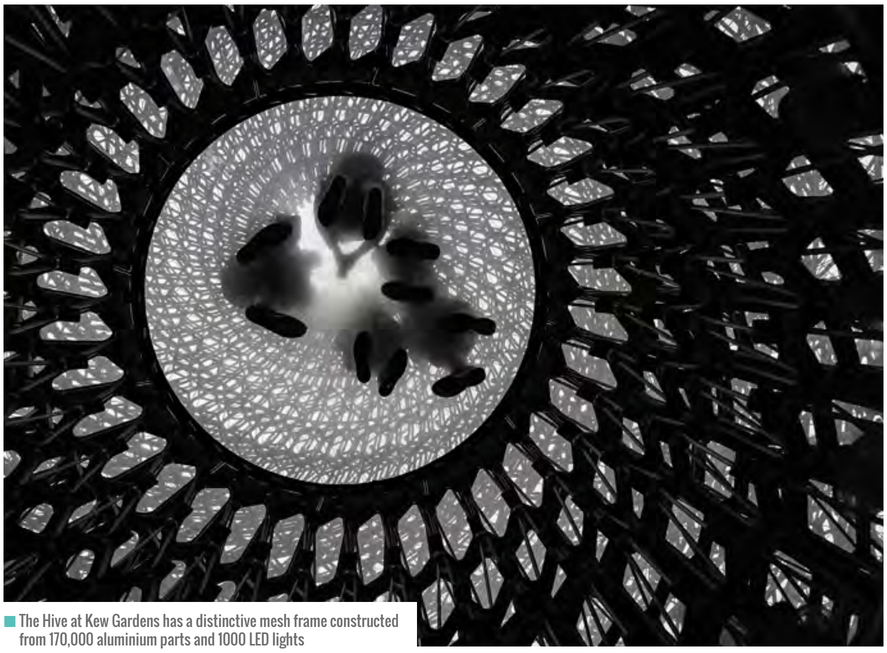
JANE WAITE

Jane has worked both as a primary teacher and in industry as a developer. She worked on the Barefoot programme, was CAS London manager, and is now a senior research scientist at the Raspberry Pi Foundation.

GED GAST

Ged is a visual arts specialist, an education consultant, and a past president of the National Society for Education in Art and Design (NSEAD).

© Miroslav Posavecc/stock.adobe.com



The Hive at Kew Gardens has a distinctive mesh frame constructed from 170,000 aluminium parts and 1000 LED lights

> THE ALGORITHMS THEMSELVES BECOME THE MEDIUM THAT THE ARTIST WORKS WITH

Algorithmic art

In interactive art, artists use computers and electronics as media, like canvases. In some works, though, algorithms play a more fundamental role, generating the art themselves.

Arguably, artists have been using algorithms for a long time — following rules as part of the artistic process. Perhaps the most stunning examples of this are the geometric images of Islamic art — and the word 'algorithm' itself comes from the Persian scholar al-Khwārizmī and his ninth-century book on the algorithms behind the Hindu–Arabic numeral system. We are taught simple algorithms in art classes at school, dating back to the Renaissance, such as those used to get perspective right. Maths, algorithms, and art have been intertwined, at least informally, for centuries.

Taking this idea further leads to algorithmic or generative art. Now, rather than the work arising purely from the creativity of an artist, it is generated by an algorithm that the artist creates. Different artworks result because the algorithm has elements of randomness in it, or because the artist tweaks the starting parameters (the inputs to the algorithm). Or, the artist may devise variations on the algorithm, exploring the different kinds of result that emerge from different changes. So in algorithmic art, algorithms themselves become a medium for the artist to work with, just like organising colour and expressive marks on a surface, painting with oils, or constructing in clay. The artist is also adding artistic judgement by deciding which images to keep and which to discard.

There are lots of places to get inspiration for the algorithms behind algorithmic art. One good source is in the algorithms of nature. A particularly creative example of this, from the University of Coimbra's CDV Lab, is a program called Photogrowth (**helloworld.cc/photogrowth**). It involves breeding a colony of virtual ants that deposit ink as they crawl over an initial image. The brighter colours of the starter picture act as food sources for the ants, and their trails lead other ants to those sources. If they don't find enough 'food', the ants die. If they do find enough, however, they reproduce, leading to more ants to continue developing the picture. Here, the parameters you can set include things like the thickness of the trails the ants leave.

Many other kinds of algorithm can be used. Our article on algorithmic doodle art in Hello World issue 9 (pages 87–89), gives some simple and unplugged algorithmic art activities for the classroom, using algorithms that mirror the developmental processes of plants. Students may find it intriguing to use these as a means of exploring the way in which our minds generate doodles and how the rules of abstraction could, or might, function. Another way is to try to create a program that codifies the way human artists work — an algorithm for human art.

The art of AARON

AARON is a robotic painting device created by the artist Professor Harold Cohen. AARON's creations have appeared in art galleries worldwide. Just as we learn the basics of art in school, AARON has been taught the rules of art, such as composition and perspective. It also knows about things in the world, such as how the parts of a person's body are connected. AARON has another kind of knowledge too: creative procedural knowledge. It knows how to follow the steps to create a painting. For example, it starts with the background of a new painting, and works its way forward to the foreground. By following these in-built rules that codify artistic knowledge, AARON creates novel

ART & COMPUTING RESOURCES

Teaching London Computing: **helloworld.cc/londonart**

National Society for Education in Art and Design: **nsead.org**

TechPathways London: **helloworld.cc/techpath** pictures. Each picture it paints is different, and it even chooses the subject to paint.

In creating AARON, Cohen made a great step forward in understanding creativity, and researchers interested in understanding computational creativity continue to pursue that mission. Ultimately, work like Cohen's is about writing programs that help us understand what it means to be human.

Computers and judgement

The next step for algorithmic art is to take the human out of the loop and let the computer itself judge which pictures are 'good'. This is a key part of creativity, and feeds back into a human's development as an artist. Great artists don't churn out lots of mediocre paintings, with a few being chosen as brilliant. They get better, and they develop a distinctive style of their own. This development as an artist goes hand in hand with the development of their judgement and so their creativity. Computer scientists working in the area of computational creativity are exploring ways for algorithms to exercise this kind of judgement.

Ant art and the Photogrowth program explore one kind of algorithm that can provide judgement: a kind of artificial intelligence algorithm called a genetic algorithm. This allows the program to take a further step, so that the computer itself judges the images it produces, and feeds that into the creative process. In this approach, the artist develops a 'fitness function' — an algorithm that gives images a score as to how well they conform to a particular aesthetic. The best images then 'breed': their parameter settings are used to create a new virtual ant colony, but with mutations — slight adjustments — to favour those winning settings. Over time, the ratings of the surviving paintings improve, as only the best survive each round, just as animal species adapt to better survive in their environment.

There are many more links between computing, maths, creativity, and art. And we would argue that the world needs more computer scientists who are trained artists, and artists who are trained computer scientists. (HW)

© Jiri Hera/stock.adobe.com

Digital photography is now the only digital art offering at many secondary schools

COMPUTING AND ART IN THE CLASSROOM

Digital technology has been used in art and design classrooms since the mid-1980s, mainly as a tool for digital drawing or manipulating and reproducing images or objects in different formats. The 1990s and early 2000s saw the greatest range of creative digital practice, with developments in digital video, animation, and even game design. Overlapping with this, England's 1999 ICT curriculum included elements of art and design in developing ideas and making things happen, for example with digital photography, multimedia, and animation.

However, recent cuts to school budgets, an increased emphasis on the computer science elements of computing, and changes to secondary examinations have had a significant negative impact on digital art and design in class. Many secondary schools have phased out the specialist software shared by art and design departments, mostly leaving digital photography as the only digital art offering. In computing departments, some new digital media courses are being introduced. In the UK, these include Creative iMedia and BTEC Creative Digital Media Production.

Some craftspeople and designers use technology as their main creative tool; others, such as the ceramicist Michael Eden, use technology alongside traditional processes and techniques. He uses CAD for rapid prototyping, and 3D printing in a plaster and gypsum material with a non-fired ceramic coating to create some of his digital forms. Multisensory work, 3D design, and generative art have become part of the art and design landscape. Famous artists such as Bill Viola use film, electronic sound, and digital images; William Kentridge combines drawing and animation; Jenny Holzer creates immersive installations of words and pictures using programming. Creative industries such as web design, interface design, games development, advertising, marketing, animation, and film incorporate digital art and design.

Have classroom practices and examination requirements kept pace with the changes in the world of art and creative media? Some people have called for young learners to have earlier access to relevant information on digital media careers, and an increased exposure to the skills needed in digital art and design professions and the creative industries. Furthermore, some suggest that increasing the computing content in art lessons could address the lack of boys opting to study art and design, while more art in computing lessons could contribute to improving the representation of girls in computer science.

Teachers in the UK can access support to increase digital art and design in schools by joining the National Society for Education in Art and Design (NSEAD) at **nsead.org**. NSEAD has a new guidance document on digital art and design. Teachers can also make use of its digital audit tool to reflect on what actions and development their school needs. When planning actions, you can find resources at Teaching London Computing (**helloworld.cc/londonart**), TechPathways London (**helloworld.cc/techpath**), and in the other ideas presented in Hello World issue 9, which focused on computing and the arts.

DEMOCRATISING DIGITAL CULTURE WITH 3D-ANIMATED FILM

3D animation still takes serious computing power, but there are ways to bring it into your classroom

One reason for the inclusion of programming in the English national computing curriculum in 2014 was the idea of democratising digital technology: we wanted people to be creators of the systems they consume. Apps, websites, computer programs, and games are built using programming, so we need to teach our children how to code.

However, coding is not the only component of digital creativity. Much of what you see in films, on TV, and in computer games has a 3D digital component, and we are also seeing growth in virtual and augmented reality technologies. If we want to democratise digital culture, we also need to enable young people to create 3D digital content. But how do we go about this?

ICT has been replaced in UK schools by computing, which focuses more on computer science. Where ICT courses might previously have had digital art components, the replacement qualifications in computer science do not. Even if a school wanted to teach 3D digital animation, maybe through its media studies course, it would require teachers with the skills to teach it, potentially expensive software, and hefty computer hardware. Today, 3D animation is one of the last areas of digital creativity that still requires serious computing power. You can program industry-standard Python through a web browser, and most programs a beginner will write can complete almost instantaneously. To create with 3D animation, though, you're going to need a fairly decent computer and a dedicated graphics card.

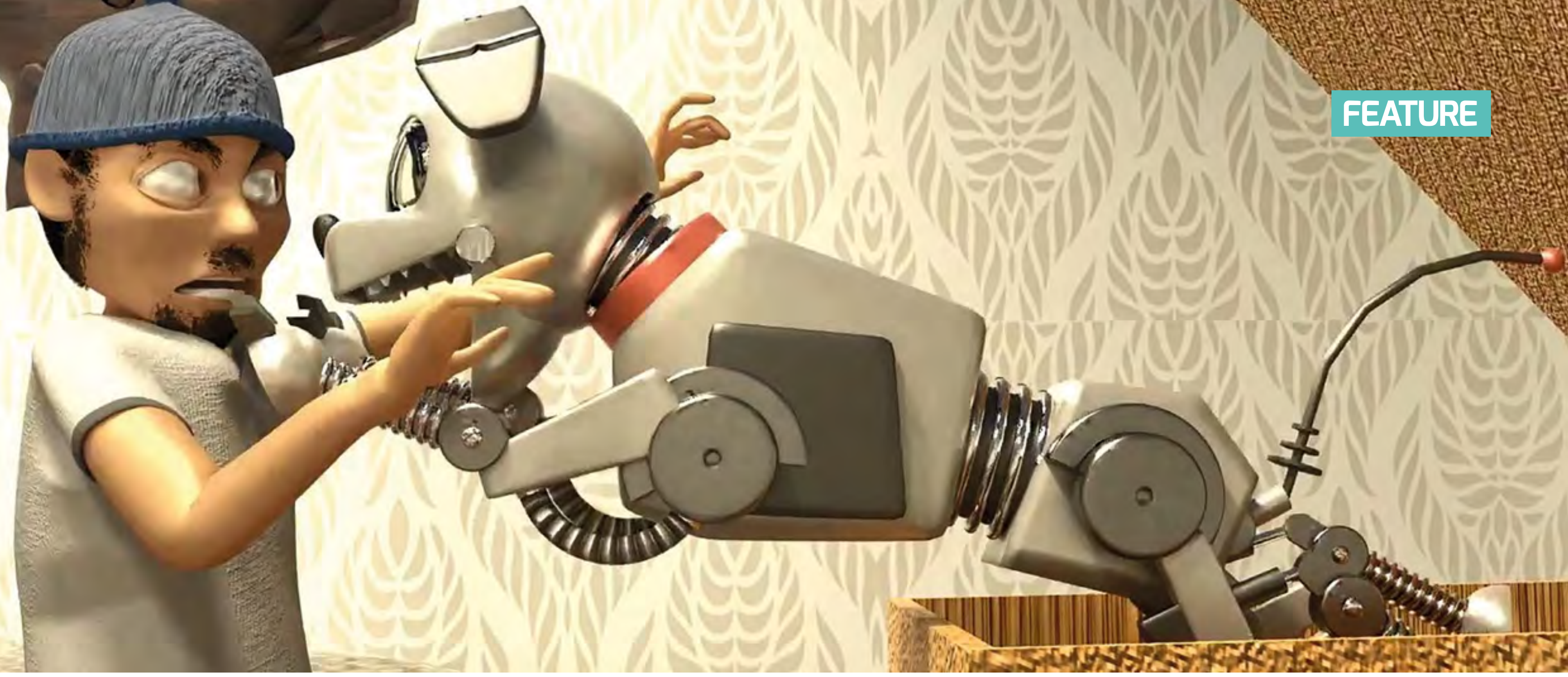
Blender 3D computer graphics software is free and used in industry

Even then, if you make short films, you'll need to wait hours, if not days, for something to render — to make each of the individual frames of a film. When Pixar's artists had finished making the characters, props, and shots, *Monsters University* would have taken 495.78 years to render on a single computer. They used thousands of computers to get around the problem, but this solution is not available to schoolchildren!

Foundations

I co-founded 3Dami (**3dami.org**) in 2012 with the intention of giving students the tools, skills, and contacts they needed to start making their own 3D digital content. I was a secondary-school teacher with students interested in getting into film, games, and animation. There was, and still is, a shortage of people working in these areas. A 2018 Nesta report, *Which digital skills do you really need?*, lists animation and multimedia production as the top areas of growth for jobs in the digital sector (**helloworld.cc/nesta2018**).

Film, games, and animation are generally quite straightforward careers to get children interested in, as they are often already avid consumers. But it's less straightforward to get students to make informed decisions about choosing these careers, based on actual experience. We wanted students to be able to gain real experience using industry-standard tools, so they could choose their future education and careers more wisely.



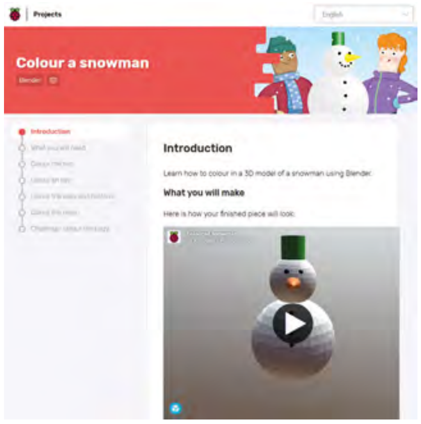
3Dami runs a seven-day camp where teams of nine students create every part of an animated short. On the first day, the students write a story for their team to work to — this means lots of paper and pens, and a complete, albeit paper-based, storyboard by the end of the day. Students then work on computers to create all the props, characters, sets, animations, and shots that make their film, with the premiere on day seven. Among the nine students, two are given roles as director and producer, with the director taking the artistic lead, and the producer using an asset management system to allocate tasks to the rest of the team and queue up the shots for rendering. We set up a 'render farm' — hundreds of computers linked together — to speed things up. A shot of seven seconds at 24 frames a second means 168 separate frames, and a normal computer might take 45 minutes to do one frame, or 5.25 days to complete the shot. If we split the 168 frames across 168 computers, the shot would be ready in 45 minutes.

> "IT'S IMPORTANT THAT STUDENTS USE INDUSTRY TOOLS TO GET A TRUE FEEL FOR WHAT IT'S LIKE TO WORK IN AN AREA

Ownership

The aim is for students to own their film and the management of it. We don't use pre-made assets, and we avoid lecturing. Students tend to learn from each other and only ask questions when people on their team can't help. Computing concepts such as modular design and computational efficiency are natural to 3D animation. One student might make a character that appears in multiple shots; another might make a bucket that's used in some of the same shots. Once these assets are linked, we have a film, and if there's an artistic change to the character's hair, it will be immediately updated in all the shots because it's linked. Learning modular design through programming is much more difficult — a buggy module will probably break a program, and a buggy haircut will just look a little rubbish. Trying to get shots rendered in time is always a problem, and students need to think carefully about how they set up their animations. Inefficient shots can bring even the fastest computer to its knees for hours at a time; in particular, smoke simulations and hair can be very computer-intensive. Someone learning how to program rarely meets a problem where they need to write more efficient code, but thinking about efficiency is a daily task in 3D animation.

It's important for students to use industry tools where possible, as this is the only way they can get a true feel for what it's like to work in an area. We base our work on the open-source Blender 3D computer graphics software (**blender.org**). Blender is free, and is used for films, TV, and games, such as the visual effects in the TV series *The Man in the High Castle*. Blender runs on old hardware that you will often find in schools. If you can't get permission to install it, it's less than a gigabyte and will run off a USB stick or shared drive.



Introduction

Learn how to colour in a 3D model of a snowman using Blender.

What you will make

Here is how your finished piece will look.

Visit the Raspberry Pi website for projects to get you started

Many of our students are now studying digital art, film, computer science, and engineering at university, and several work in the games and film industries. 3D digital art is a skills shortage area that many students love! If you'd like to get started, please check out the materials we've made for schoolchildren over at **b3d101.org** and **helloworld.cc/blenderprojects**. (HW)

PETER KEMP
Peter is a senior lecturer in computing education at King's College London, UK. He taught computing in secondary schools through the Teach First scheme and set up 3Dami in 2012.

SEPARATING THE LEARNING FROM THE APPLICATION

Josh Crossman explores the importance of leading teaching with concepts and skills, rather than the nuances of software applications

Computing is a broad discipline, rich in concepts and skills, which can be taught through many different technologies and software applications. As we have developed units of work for the Teach Computing Curriculum (TCC) (helloworld.cc/tcc), we have reflected on how best to support learners in developing lifelong understanding and skills. An important aspect of this is separating the learning from the applications used to teach it (such as software packages and programs) and instead, leading with concepts.

This principle is of particular relevance in the 'Creating media' strand of the TCC, where learners select and create a range of media including text, images, sounds, and video. It is imperative that learners can use their knowledge and understanding more widely, rather than getting lost in the nuance of particular applications.

Scaffolding conceptual understanding

Leading with concepts is, of course, a consideration to make more broadly when teaching. In literacy classes, for example, learners might write a diary entry, using either pencil and paper or a word processor. The learner may be proficient at using either medium, but that doesn't mean they can write a good-quality diary entry! They have to learn how to structure it, as well as the importance of writing in the first person and the use of appropriate vocabulary.

Similarly, when teaching computing, you generally need a tool or software application to bring concepts to life. Which software application you choose will depend on a number of factors, such as accessibility for learners; the time it takes to understand how to use it; and whether it is a free or a paid-for product. The key thing to recognise is that the application should be used to scaffold conceptual understanding, rather than being an integral part of the learning.

Benefits

By following a more application-agnostic approach to teaching computing, we can achieve several benefits:

Transferable knowledge

With such a range of digital applications available, it is important that learners develop knowledge that can be easily transferred. If educators lead with concepts rather than tools, learners will be able to reapply their knowledge to other applications and technologies in their lives. A simple example is understanding the copy and paste function. It is a key concept that, once understood, can be applied to most other applications and programs.

Learner independence

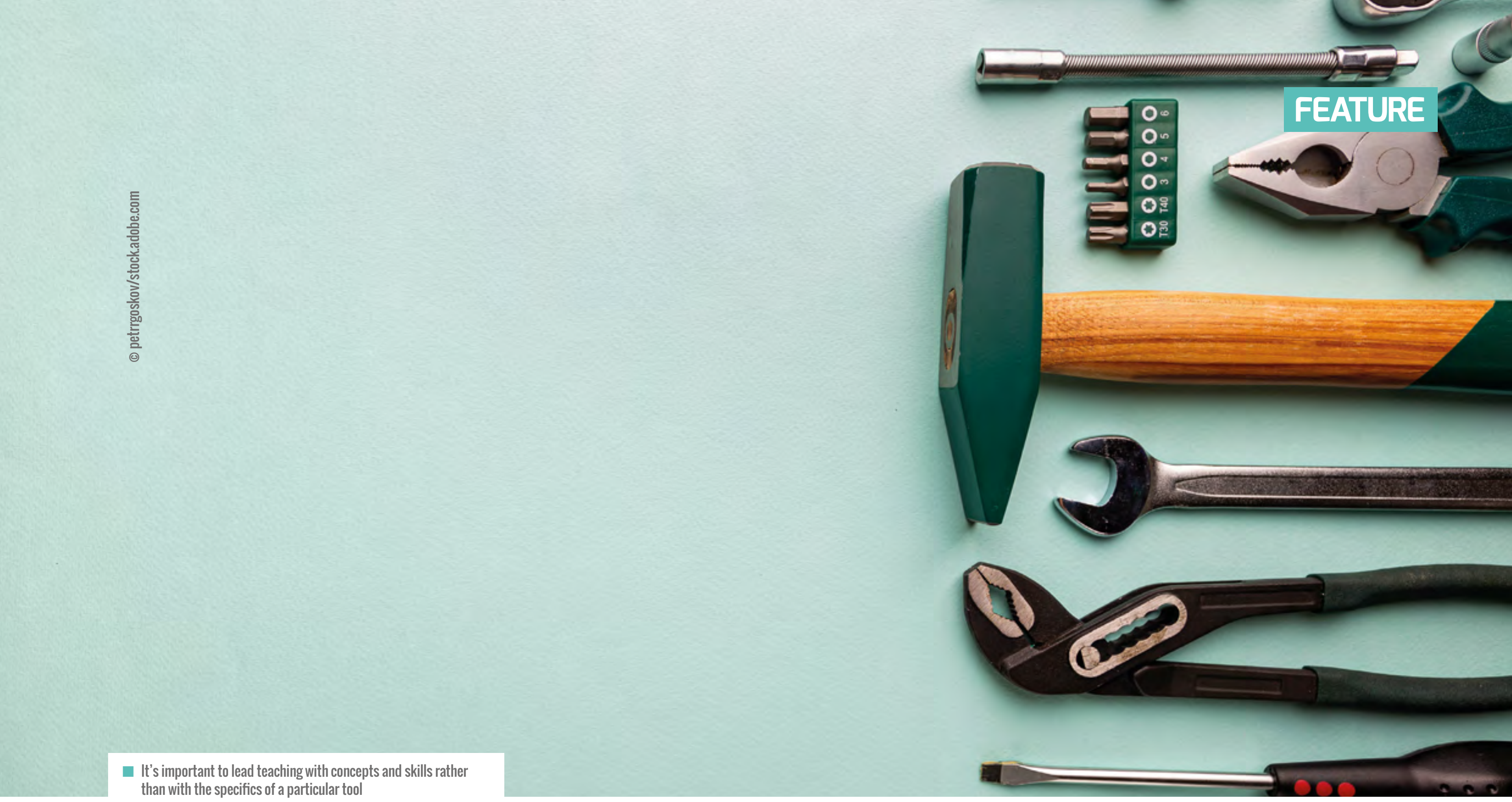
As learners become familiar with how to accomplish certain tasks, they become more independent. This enables them to traverse different applications more easily as they progress. When learners need to create more complex media, for example, they will need to use an application with more complex features. In the TCC, this involves learners progressing from basic vector drawing applications, such as Google Drawings in Year 5 (aged 9–10) to more complex applications such as Inkscape in Year 8 (aged 12–13). With an understanding of key concepts and skills, learners can apply this knowledge with less support, allowing educators to focus on embedding new concepts and understanding instead.

Software variations

Throughout the TCC 'Creating media' units, we have chosen software applications that are free, accessible, and learner-friendly. However, the majority of these applications are web-based, so require the internet. Whether a school has a strong enough internet connection will be a factor in deciding whether that application is suitable, or whether a locally downloaded application would be more beneficial. This is a very specific use case, but developing units of work that aren't focused on the specifics of a tool makes it much easier to transfer lessons to other applications if necessary.

Teacher questioning

The questions we ask as educators are arguably one of the most important aspects of teaching and learning. When teaching learners processes to achieve something within a piece of software, such as how to draw a line, the only understanding you can ask of those



It's important to lead teaching with concepts and skills rather than with the specifics of a particular tool

learners is to repeat the process back to you. This is low-level understanding, relying solely on their memory. If, instead, you teach them to recognise the familiar line icon, make connections with how they apply other tools such as the shape tool, and encourage learners to consider prior knowledge, your questions can be deeper, such as: "How can I create a line to add to my drawing?" and "How did you know to choose this specific tool?"

The benefits of this approach are numerous, but it can be daunting for a teacher initially. If learners are used to being told every icon to click, you may have to transition to a more open approach by using games, such as making learners detectives who have to search for clues as to what different icons can do. The effort will be worth it, though — I promise!

THE KEY THING TO REMEMBER IS THAT APPLICATIONS SHOULD SCAFFOLD CONCEPTUAL UNDERSTANDING

Assessment

Changing how you approach teaching and learning when making use of software can also have an impact on assessment. There are new questions to consider, such as: How much should we assess a learner's ability to use the specific functions of a tool? What if they achieved the correct end goal, but in a convoluted way? Having a clear focus on the conceptual understanding rather than the process of using a tool or application allows greater clarity when assessing learning.

Throughout the 'Creating media' strand of the TCC, we use rubrics as possible summative assessment tools for educators to use. We have tried to ensure these rubrics are always focused on the concepts and skills introduced in the learning, rather than on the nuances of an application. In the Year 5 vector-drawing unit (**helloworld.cc/y5vector**), for example, learners are assessed on whether they can move objects to different layers in the drawing in a suitable way that fulfils the required task. To achieve this, learners will first need a conceptual understanding of what layering is in vector-drawing software; they will then need to identify the process for ordering objects in the layers; and they will finally need to choose a suitable scenario in which they could make use of their layering knowledge. It will be evident from the learner's final product where they have used this skill and whether they have understood the concept, enabling a more focused assessment of their holistic understanding.

As with any subject, we want to maximise learning time and ensure learners can apply concepts and skills to other areas of their learning. Removing the specifics of the software application allows us to do this, and ensures the teaching and learning experience is driven by the learner's needs and not by the application used. (HW)

JOSH CROSSMAN
Josh is a programme coordinator at the Raspberry Pi Foundation, working across programmes such as the Teach Computing Curriculum and Hello World. He is a Raspberry Pi Certified Educator and a former primary teacher.

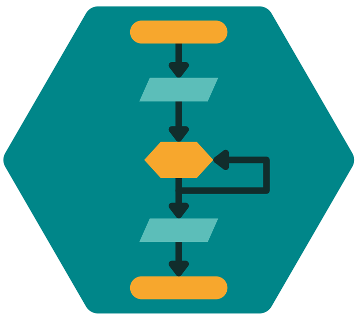
ALGORITHMS AND DATA STRUCTURES

Algorithms and data structures enable learners to explore and express core programming concepts in a more abstract manner, independent of a specific programming language. In this way, algorithms and data structures are useful for planning, expressing, and comparing the intended operation of a computer program.

From an early stage, learners will plan and represent their programming plans in a variety of ways, including designing sketches and simple pictorial or symbolic algorithms, eventually progressing to more formal approaches, including flow charts and pseudocode. At the upper stages of school education, learners will begin to use standard algorithms (such as those for sorting, searching, and route finding) to make comparisons between different approaches and develop an understanding of algorithmic complexity. Later in their education, learners also typically study abstract and complex data structures, such as stacks, queues, and trees, as well as the algorithms used to manipulate them.

IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** algorithms and data structures, in summary
- **What the research says:** the power of pseudocode
- **What the research says:** computational thinking and personality
- Learning to program with Fakebots
- Teaching abstraction skills



IN SUMMARY

ALGORITHMS AND DATA STRUCTURES

Comprehend, design, create, compare, and evaluate algorithms

STAGE 1

- Identify that algorithms are clear, precise steps to complete a task
- **Recognise the role of algorithms as part of program design**
- Decompose a simple task into two smaller tasks
- **Express algorithms through symbols, simple sketches, and in written form**
- Adapt a template or example to plan an algorithmic solution

STAGE 2

- Describe how algorithms relate to program design
- **Distinguish between a program and an algorithm**
- Recognise that different algorithms can achieve the same outcome
- **Follow an algorithm in order to predict the outcome and identify errors**
- Decompose a task into several smaller tasks
- **Choose appropriate formats to express algorithms and designs (sketches, flowchart symbols, text)**
- Design solutions that reflect the capabilities and constraints of the intended system

In the table below, you will find learning outcomes associated with the 'Algorithms and data structures' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Describe how algorithms relate to different parts of a programmed solution	Recognise factors that affect the efficiency of searching and sorting algorithms	Describe a range of standard algorithms including searching, sorting, graph traversal, and shortest path
Distinguish between real-world scenarios and computational models	**Describe how common searching and sorting algorithms work**	**Describe how the complexity of an algorithm is measured in terms of time and space requirements**
Walk through an algorithm recording the value of variables	Use trace tables to walk through algorithms	Distinguish between problems that are tractable, intractable, and unsolvable
Analyse algorithms and suggest potential improvements	**Compare algorithms and select the most efficient for a given scenario**	**Analyse standard algorithms and express their efficiency using big O notation**
Express algorithms using text and flowcharts	Explain the need for abstraction and decomposition when planning a program	Compare the efficiency and suitability of algorithms for different scenarios
Independently decompose a task into smaller tasks, events, and actions	**Use formal flowchart symbols to communicate algorithms with others**	**Represent and convert algorithms using pseudocode, flowcharts, and structured English**
Use abstraction to highlight key parts of a task when designing a solution	Use a shared pseudocode to communicate algorithms with others	Use data structures and abstract data types to organise and manipulate data effectively
Write algorithms for whole solutions or individual components	**Write algorithms that manipulate data structures such as lists, arrays, and records**	**Explain the difference between static and dynamic data structures**
		Design the main parts of an application including the interface, required data, and key algorithms and data structures

IN DEFENCE OF PSEUDOCODE

STORY BY Eirini Kolaiti

When I used to teach programming, it didn't take me long to realise how easily my students could confuse or even forget basic structural elements of a programming language. They would be tripped up by misusing capitalisation, white space, or parentheses. Basic syntax errors such as these would sidetrack students from focusing on developing a solution. Instead, they would resort to trying out every possible instruction they could find online in the hope that it would solve their errors. It turns out that my experiences weren't unique, and are also supported by research in computing education.

The case for simple languages

In 2006, researcher Linda Mannila and her colleagues compared 60 programs written by students aged 16–19 after their first programming course, in either Python (a simple language, developed for its readability) or Java (a more advanced language). They analysed the assignments, both in terms of syntax and logic errors and their overall functionality (**helloworld.cc/mannila2006**).

The results were remarkable. Programming in Python not only helped students avoid making syntax errors, but also allowed them to solve the given problem. The percentage of Python programs that ran correctly and fulfilled the intended purpose was more than double that of the Java programs.

One potential drawback of teaching programming in Python rather than a more complex language, such as Java, is that it could do students a disservice if they then needed to relearn aspects of programming when moving to a less intuitive language, like Java. However, the team did not find this to be the case: students who had first learnt to program with Python were at no disadvantage when switching to Java.

So if choosing a simple language can play such a positive role in student progression, why not opt for that option?

From simple to pseudocode

Researchers Allison Elliott Tew and Mark Guzdial took the matter of programming language choice even further (**helloworld.cc/tew2011**). In 2011, they developed a way of comparing the knowledge of university students who took introductory programming courses in Java, MATLAB, and Python. They wanted to make the assessment language-independent, so they used pseudocode instead of any of the taught programming languages. Pseudocode makes use of simple English to describe what a program does. It is laid out in a similar manner to a programming language, but removes some of the clutter that is needed for a machine to understand the code — clutter that increases the complexity to a human reader.

In a 2006 study, students who had first learnt to program with Python were at no disadvantage when switching to Java

The results demonstrated that a pseudocode-based assessment can accurately determine students' programming competency, regardless of their programming background. This means that students could transfer their comprehension of fundamental programming concepts to pseudocode notation. Reversing this logic, surely we can use pseudocode to scaffold the learning of programming concepts.

Walk before you run (a program)

In 2004, an international group of researchers, led by Raymond Lister at the University of Technology in Sydney, conducted a study regarding programming competency across seven countries (**helloworld.cc/lister04**). Instead of asking students to produce their own programs, the researchers examined whether students could understand existing code (written in Java or C++) by predicting the output of a given program. The results suggest that what stops many students from performing well in programming tasks is not a lack of ability to problem-solve, but a fragile knowledge of fundamental concepts. Students were unable to hand-trace code (where the values of variables are calculated by hand) because of an insufficient command of basic programming tasks, such as iterating over an array or the use of recursion. These areas are mostly related to an ability to read code rather than write it.

From these pieces of research, one could argue that as educators we should use programs in pseudocode to foster these preliminary skills, so that students benefit from a reduction in the cognitive load caused by language-specific syntax. After all, most programming languages are not designed with the aim of teaching programming, whereas pseudocode can be adapted to meet the needs of the students.

Pseudocode activities can be used in lessons in order to practise reading and tracing code, and as an opportunity to discuss basic programming concepts.

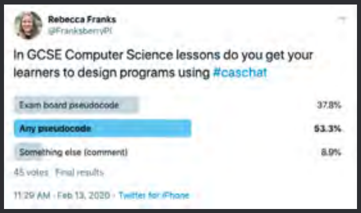
“ PSEUDOCODE LOOKS LIKE A PROGRAMMING LANGUAGE, BUT IT REMOVES SOME OF THE CLUTTER SO IT'S EASIER TO UNDERSTAND

>

PSEUDOCODE AT GCSE

REBECCA FRANKS
Learning manager at the Raspberry Pi Foundation

I asked upper-secondary computer science teachers on Twitter which type of pseudocode they used for designing programs. 38 percent followed their exam-board-specific pseudocode and 53 percent did not specify a syntax. Several commented that it was more important that learners could effectively design their own programs and that the syntax used was less relevant.



Rebecca Franks
@FranksberryPi

In GCSE Computer Science lessons do you get your learners to design programs using #caschat

Exam board pseudocode	37.8%
Any pseudocode	53.3%
Something else (comment)	8.9%

45 votes · Final results

11:29 AM · Feb 13, 2020 · Twitter for iPhone

In UK GCSE qualifications (for ages 14-16), there is currently no requirement for students to use a specific pseudocode syntax. However, most exam boards have their own unique approach to pseudocode for presenting questions. Students need to be familiar with these to answer questions successfully. Exam questions will indicate the form or response required, whether it's a specific programming language, natural English, pseudocode, or a flowchart.

PSEUDOCODE ACTIVITIES

Here are some ideas on how to incorporate pseudocode into your teaching:

- Start activities with pseudocode to discuss concepts before converting into code to deal with syntax errors, to test/debug the algorithm, and to check the algorithm's structure. It should be easier to write an algorithm after students have worked on a pseudocode version, rather than starting with a programming language.
- Ask students to write a program using pseudocode and then swap with a partner for them to write it in a programming language.
- Give snippets of pseudocode to test basic misconceptions, for example in the use of recursion.
- Give small programs that students can hand-trace, writing out the values of variables as the program progresses, to check they understand the code.

Linear search in pseudocode:

```
function linear_search(list, element)
    for i = 0 to len(list) - 1
        x = list[i]
        if x == element then
            return TRUE
        endif
     next i
     return FALSE
endfunction

v =  ['Bob', 'Doug', 'Alice']
value = input("Enter search string or q to quit: ")
WHILE value != ‘q’ AND value != ‘Q’
    print( linear_search(v, value) )
    value = input("Enter search string or q to quit: ")
ENDWHILE
```

Linear search in Python:

```
def linearSearch(list, element):
    for x in list:
        if x == element:
            return True
    return False
v = ['Bob', 'Doug', 'Alice']
while True:
    value =input("Enter search string or q to quit: ")
    if value.lower( ) == 'q':
        break
    else:
    print linearSearch(v, value)
```

Students could then convert the pseudocode programs (their own or each other's) into compilable code. This gives them the opportunity to experiment with how to implement algorithmic constructs using the specific features of a programming language, and how to deal with syntax errors. Having the stepping stone of pseudocode therefore helps with the skills needed to test and debug algorithms.

Pseudocode can also be used for formative assessment. Low-stakes tests and starter activities that use pseudocode snippets to test specific misconceptions can help unpack the underlying processes of program execution. I have found that challenging students with small, targeted pseudocode programs helps with engagement and information retention. In this way, pseudocode could pave the way towards gaining fundamental knowledge and skills through reading and tracing code before moving on to writing actual programs and dealing with the inevitable corollary of syntax errors. (HW)

FURTHER READING

- ✓ Mannila, L., Peltomäki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education, 16*(3), 211-227. **helloworld.cc/mannila2006**
- ✓ Tew, A., & Guzdial, M. (2011). The FCS1: A language independent assessment of CS1 knowledge. *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 111-116. **helloworld.cc/tew2011**
- ✓ Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin, 36*(4), 119-150. **helloworld.cc/lister04**

Pseudocode activities can be used in lessons to practise reading and tracing code

Linear search in Java:

```
import java.util.Vector;
    public class JavaEx {
        public static boolean linearSearch(Vector v,
Object o) {
            for (int i=0; i 5 v.size( ); i++) {
                if (v.elementAt(i).equals(o)) {
                    return true;
                }
             }
             return false;
     }

    public static void main(String args[ ]){
        Vector v = new Vector( );
        BufferedReader in = new BufferedReader(
            new InputStreamReader (Systems.in)
            );
            String[ ] values = {"Bob", "Doug", "Alice"};
            for (int i = 0; i 5 values.length; i++) {
                v.addElement(values[i]);
         }
            String value;
            while (true) {
                System.out.println("Enter search string
or" + " q to quit: ");
            value = in.readLine( );
            if (value.toLowerCase( ).equals("q")) {
                break;
            }
            else {
                System.out.println(linearSearch(v,
value));
            }
        }
    }
}
```

Linear search in Python, Java, and pseudocode from Mannila and colleagues' 2006 research. *Reprinted by permission of the publisher (Taylor & Francis Ltd,* ***tandfonline.com****)*

RESEARCH
FRIDAYS FOR FUTURE!
FURTHER READING
ACTIVITIES TO TRY IN YOUR CLASSROOM
THERE IS NO CURRICULUM THAT GIVES PUPILS AN OVERVIEW OF THE EFFECTS OF CLIMATE CHANGE ON SOCIETY

TEACHERS' VIEWS ON CLIMATE CHANGE EDUCATION

STORY BY Bonnie Sheppard

Fridays for Future, Extinction Rebellion, the fossil fuel crisis — climate change and sustainability are hot topics that increasingly affect and interest students of all ages. A recent study of teachers in England showed that nearly all teachers had noticed an increased eagerness to discuss climate-related issues in their classes, especially in primary education. However, the national curricula of most countries confine climate change education to science and geography lessons in secondary school. Teachers of other subjects and those with less training in climate science therefore often feel ill-equipped to address their students' questions or incorporate climate- and sustainability-related examples into their learning modules. The study also suggests that empowering teachers across all subjects to deliver climate education could solve many shortcomings of the current approach, and also improve the dialogue about climate change and sustainability.

Earlier, cross-curricular, better funded

If there is one key takeaway message from research into climate change education in the past few years, it's that there is currently

TO SUBSCRIBE VISIT:

helloworld.cc/subscribe

(Hello World)
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
TOMORROW'S ENGINEERS
The first Schools COP
PHILOSOPHY FOR CHILDREN
A fresh approach to digital literacy
INTERVIEW: CENTRE FOR AI & CLIMATE
AI for mitigating climate change
Issue 19
Jun 2022
helloworld.cc
SUSTAINABILITY & COMPUTING
HOW CAN TECHNOLOGY BE GOOD FOR THE PLANET?
PLUS
CLOZE QUESTIONS • RESEARCH CENTRE LAUNCH • REPAIR CAFÉS • COOLEST PROJECTS
COSPACES • UPGRADE CULTURE • ONLINE SAFETY • COMPUTING QUALITY FRAMEWORK
BAKING IMPOSSIBLE • BROWSER-BASED CODE EXERCISES • AI & ENVIRONMENTAL RISKS

Not a UK-based educator?

- Buy a print copy from **helloworld.cc/store** — we ship to over 50 countries
- Subscribe to receive the free PDF on the day it is released
- Read features and news at **helloworld.cc**

FREE PDF
for anyone, anywhere

HEAR FROM THE WRITERS!

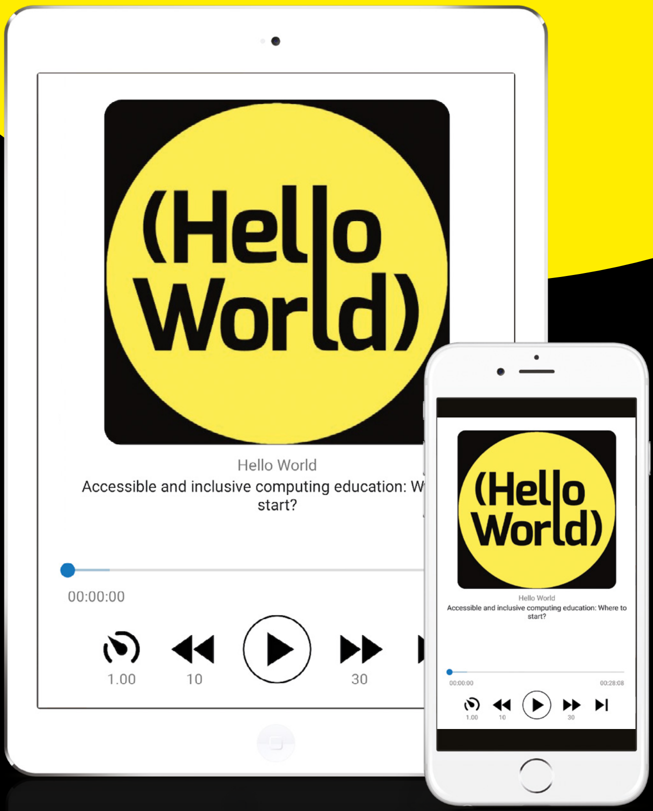
Listen to Hello World's podcast now:

- Get more great Hello World content between issues
- Hear directly from the educators behind our articles
- Listen on the move — delve a little deeper and have some fun along the way

Subscribe to the podcast!

- Apple Podcasts **helloworld.cc/applepod**
- Spotify **helloworld.cc/spotifypod**
- Google Podcasts **helloworld.cc/googlepod**
- Amazon Music **helloworld.cc/amazonpod**

SCAN ME


Hello World
Accessible and inclusive computing education: Where to start?
00:00:00
1.00
10
30

OUR MOST POPULAR EPISODES

How moral is your machine? Ethics in computing education

Accessible and inclusive computing education: where to start?

How can we get everyone excited about code?

COMPUTATIONAL THINKING, CONFIDENCE, AND PERSONALITY

STORY BY Lucia Flóriánová

Computational thinking is a topic that's widely discussed within computing education. It's a key thinking skill that students need to develop if they are to live in a technology-driven society.

Computational thinking is generally understood as a type of logical thinking that can help students to problem-solve, both with and without the use of computing devices. It relates to children's cognitive abilities, such as their reasoning, spatial, numeracy, and problem-solving abilities.

More than cognitive ability

Román-González and his colleagues, however, concluded in their 2017 study that cognitive abilities weren't the only link to computational thinking skills: they studied how computational thinking is also linked to the non-cognitive factors of self-efficacy and personality. Quantitative analysis suggests that only 27 percent of computational thinking is explained by cognitive factors. Almost the same proportion — 24 percent — is related to non-cognitive factors, mainly self-efficacy, openness to experience, conscientiousness, and, surprisingly, extraversion.

The study found a possible link between students' computational thinking and their self-efficacy — that is, students' perceptions of how well they performed. It appears that if learners believe they can perform well, they're likely to achieve better results. Lower self-efficacy in girls could also help to explain the gender gap in computing.

Computational thinking also seems to be linked to openness and conscientiousness. This finding doesn't come as a surprise, as these are the aspects of personality that are most closely related to academic performance, dealing with open-ended problems, and persistence in working on difficult tasks.

A positive link between computational thinking and extraversion, on the other hand, contradicts most existing research, and challenges stereotypes. The consensus used to be that brilliant programmers are usually introverted, but this seems to be changing. Computing is becoming less of a solitary activity, and now involves more social interaction and collaboration.

Improving computational thinking

These findings provide new insight into how we should approach computing education. Although computational thinking is a cognitive psychological process, it is significantly influenced by non-cognitive factors.

To make children think computationally, educators should therefore focus on their development beyond cognitive abilities, such as boosting confidence in programming. It would be especially useful to improve children's self-efficacy in specific computing tasks, and helping children see their success through concrete examples can be one way of making them learn more. Finally, it is paramount to teach computing in an environment that welcomes different types of personality and is freed from stereotypes about a typical computational thinker. (HW)

FURTHER READING

- ✓ Román-González, M., et al. (2018). Extending the nomological network of computational thinking with non-cognitive factors. *Computers in Human Behavior, 80*, 441-459. **helloworld.cc/gonzalez2018**

A pupil uses a Bee-Bot: can you spot the Barefoot Fakebot to the side?

FROM FAKEBOT TO BEE-BOT

Jane Waite and **Pam Popay** share how laminated card representations of popular floor robots can help students learn to program

As a resource to help pupils learn to program, the Bee-Bot floor robot has been a success in primary classrooms and clubs in the UK. A present-day simplified version of Seymour Papert's Logo turtle robot, the plastic device looks like a 20cm bee with a smiling face, bright colours, and simple buttons on its back to control its motion. Over three quarters of a million units have been sold by the education providers TTS since the product's launch in the early 2000s, with other similar products on the market also being used by educators.

Each Bee-Bot has a left and a right button, and straight on and reverse buttons, which can be combined to write a program which, when run, causes the robot to move around the room. There is also a 'Clear' button and a 'Go' button. The 'Go' button executes the sequence of commands that were most recently entered since the last time 'Clear' was pressed. The 'Clear' button wipes the device's memory, and if it's switched off, the memory is also cleared.

The birth of Fakebots

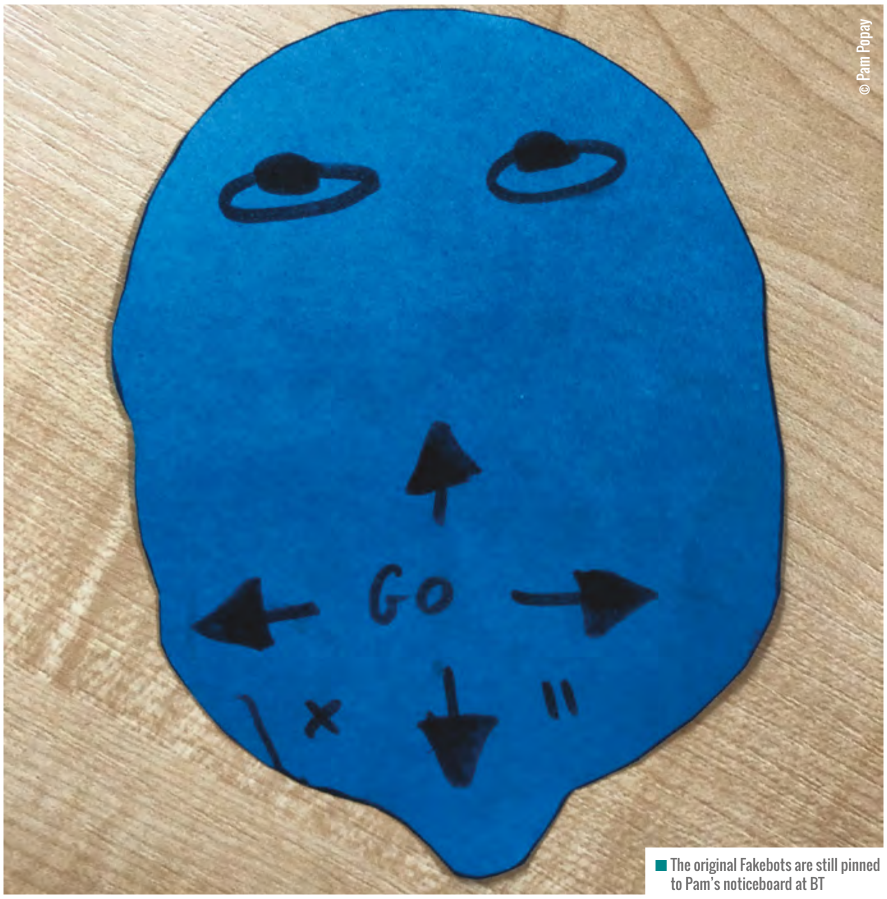
In 2013, Pam Popay and a colleague at BT were spending a lot of their time developing and trialling resources to support the newly proposed computing curriculum in schools in Suffolk, UK. When they discovered that all the schools they were visiting had Bee-Bots, it seemed obvious that they should develop some resources to be used with the devices which could support the delivery of the new intended computing curriculum across different year groups.

The BT team started with 'having a go' sessions that introduced children to the Bee-Bots. Rather than randomly playing with the device, pupils were encouraged to think like scientists and examine what each button did. By doing this more guided activity, the students gradually discovered the functionality of the Bee-Bot.

As the team from BT developed more tasks, they realised that students needed a way to record their planned sequence of instructions, as it was hard for children to debug their intended instructions without it. After experimenting with different methods, they introduced a set of arrow cards along with 'Go' and 'Clear' cards. The children used the cards to construct a command list, which they could then test. Finally, when students thought the sequence was correct, they were then given the Bee-Bot to program. However, it soon became clear that the physical robot was a distraction from the

© BT and BCS Barefoot

Barefoot Fakebots are available on the Barefoot website, but you can get your children to make their own



© Pam Popay

The original Fakebots are still pinned to Pam's noticeboard at BT

thinking process and that pupils wanted to use trial and error instead of thinking ahead to work out the solution to the problem.

During a lesson break, Pam had an idea. She found some card in the classroom, drew around a Bee-Bot, and added buttons and a face. She then introduced this paper-based representation of a Bee-Bot to the class. Pupils were now asked to complete activities with the card bee, while the Bee-Bots rested safely in their beehive. One of the students in that class called the card bee a Fakebot, and this is what they have been called since.

Pupils used the Fakebot to test their planned sequences of commands. They worked in teams: one pupil was the sequence designer and could organise the sequence of commands using the arrow cards, and another pupil controlled the Fakebot, using it to test the arrow card command sequence. Once the team thought all was well, they could collect a real Bee-Bot, and a third member of each team, the programmer, entered the commands into it. The group then waited to see what happened. During this time, they followed the arrow card sequence as the Bee-Bot moved around.

Through working with Fakebots, Pam has recognised several essential features that can impact on their successful use. Firstly, they must be the correct size, so that their tested movements are relatively accurate. Secondly, depending on the activity, it can be useful to have Bee-Bots with distinguishing features, such as different eyes, so that each group knows which Fakebot is theirs. Thirdly, if required, a Fakebot can have L (left) and R (right) written on them to help children recall vocabulary for discussing movement.

Fakebots as a widespread resource

Around the time she was introducing Fakebots in teaching resources, Pam shared her findings about the Fakebot with Jane Waite, who was very excited by the idea. Both Pam and Jane were involved in the development of the Barefoot Computing programme at the time, and included Fakebots in the Barefoot resources. Set up by BT and Computing at School, Barefoot helps to empower primary-school teachers across the UK to deliver the computing curriculum brilliantly with free and engaging lesson plans, online guides, and workshops.

It would be fantastic to hear how educators are using Fakebots in their classrooms and how they affect pupil learning. Please share your experiences with us on Twitter (@janewaite). (HW)

JANE WAITE & PAM POPAY
Jane is a senior research scientist at the Raspberry Pi Foundation. Pam works on education engagement strategy at BT.

FAKEBOT RESOURCES

You can find Fakebot activities, such as Barefoot's *Bee-Bot Basics Activity* and *Bee-Bots 1, 2, 3 Programming*, at **helloworld.cc/beebot1** and **helloworld.cc/beebot2** along with other downloadable resources that help bring computing to life in the classroom.

Bee-Bots and Fakebots are also used in resources from the Teach Computing Curriculum (**helloworld.cc/tcc_robot**) and TTS (**helloworld.cc/tts**).

FEATURE

ABSTRACTION: THE IMPORTANT BITS

Matthew Parry discusses the concept of abstraction and shares ideas on how to teach abstraction using the PRIMM approach and My Blocks in Scratch

The Collins English Dictionary defines abstraction as "the process of formulating generalised ideas or concepts by extracting common qualities from specific examples" or "something which exists as a general idea rather than as an actual example". In other words, it is about removing complexity in order to increase understanding, without losing the core message.

We encounter abstractions around us every day. They can be shortened versions of something, to give an example of the whole thing. For example, the British television programme *Match of the Day* shows only highlights of football matches; music websites allow you to stream excerpts of upcoming albums; and the blurb on the back of a book gives only an indication of the whole story.

Abstractions can also be symbols that are used to convey meaning while reducing complexity. Examples are road signs, the London Tube map, class timetables, and infographics.

We also regularly entertain abstract ideas in our day-to-day living. For example, most of us have only a conceptual understanding of most of the machines we use. We use the terms 'car', 'microwave', and 'mobile phone' without really knowing what they do or having any concept of how they work.

Word problems in mathematics are a great example of abstraction. Pupils need to choose (abstract) the relevant information from the description in order to create the number sentence that solves the problem.

Abstraction in computer science

Within computer science, abstraction is one of the key elements of computational thinking. Barefoot Computing states that "abstraction is about simplifying things — identifying what's important without worrying too much about detail", while educational researcher Keith Turvey says that "abstraction is a process by which any unnecessary detail is omitted in order to help us to solve a problem or achieve a specific outcome".

One of the aims of England's National Curriculum for Computing is to ensure that all pupils "can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms, and data representation".

Abstraction allows us, as computer scientists, to concentrate on the important bits of developing a solution by ignoring the irrelevant detail — much as pupils do for maths word problems.

For example, we may have been asked to create an algorithm for making a cup of tea. In our algorithm, we can ignore some of the complexity that this task would entail, such as how to use a kettle, and use (abstract) instructions such as 'switch the kettle on' in order to make our algorithm simpler and easier to understand.

Abstraction in Scratch

The majority of programming languages are abstractions. They hide the complexity and, for us humans, the incomprehensibility of the binary instructions that computers use. A lot of the keywords and commands within programming languages, for example the print and input commands, also abstract more complex tasks.

Within Scratch, there are a number of blocks that abstract the processing that is being performed under the hood. **Figures 1 and 2** below show two such blocks, and you can see the steps that are hidden from the user in their captions.

Abstraction lesson idea using Scratch

Using decomposition and the PRIMM (Predict–Run–Investigate–Modify–Make) pedagogical approach to structuring

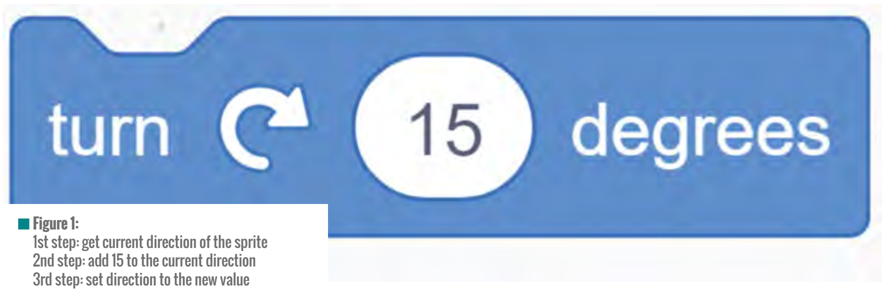


Figure 1:
1st step: get current direction of the sprite
2nd step: add 15 to the current direction
3rd step: set direction to the new value

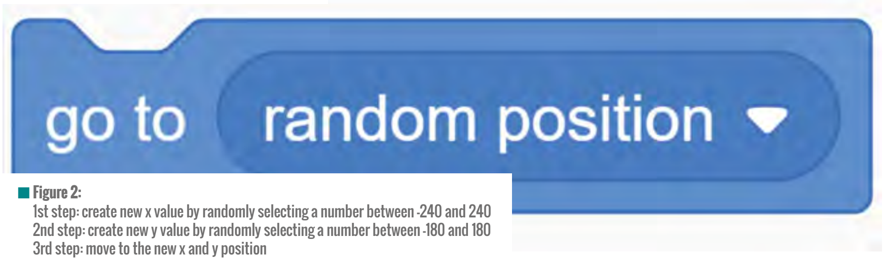


Figure 2:
1st step: create new x value by randomly selecting a number between -240 and 240
2nd step: create new y value by randomly selecting a number between -180 and 180
3rd step: move to the new x and y position

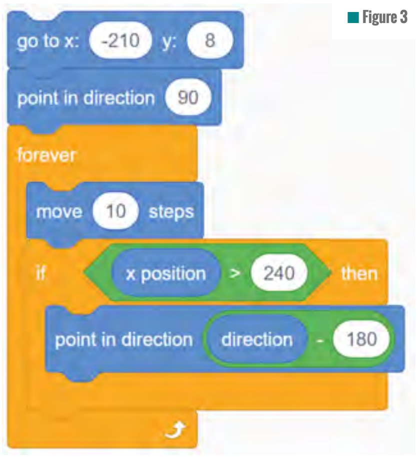


Figure 3

programming lessons (see the boxout for more details), we can unpick another of the Scratch blocks to begin to understand how abstraction works.

Take your class through the following steps, referencing **Figure 3**:

Predict: ask your students to look at the code and ask them, "What does this code snippet do? What will happen to the sprite?"

Run: get your students to run the program and ask them, "Does the code do what you thought it would? If not, can you work out why?"

Investigate: ask your students to label and comment on the code. You can ask questions such as, "What does each block do? What do the numbers represent? What happens if you change the numbers?"

Modify: ask students to alter the code to make the sprite bounce on the left-hand edge of the screen too.

Make: get students to add code to make the sprite bounce at the top and bottom of the screen too. Ask them what they will have to change to test this code. For an added challenge, ask students to try and replicate the **if on edge, bounce** block.

Abstraction using My Blocks

Using My Blocks in Scratch is an excellent way to develop understanding of abstraction. In creating an abstraction, pupils must fully understand what information needs to be hidden from the user.

One of the complexities of teaching Scratch to younger children is the need for them to understand coordinates and the use of negative numbers to control the movement of a sprite around the stage. We can use abstraction to hide those complexities by creating new blocks to move left, right, up, and down. Take students through the following steps:

Predict: what do you think the block in **Figure 4** will do?

Run: using the **when right arrow key is pressed** block, run the **Move Right x steps** block. Does it do what you thought it would?

Investigate: investigate the block definition in **Figure 5**. How does it work? Will it work for all directions?

> TO CREATE AN ABSTRACTION, PUPILS MUST UNDERSTAND WHAT INFORMATION NEEDS HIDING FROM THE USER

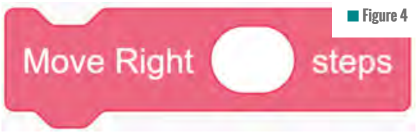


Figure 4

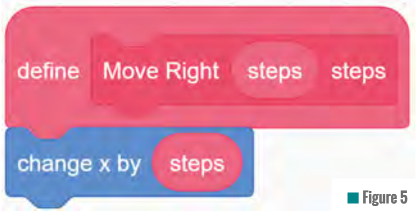


Figure 5

Modify: can you create other blocks for moving left, up, and down?

Make: can you create another block called **stay on screen**, to stop the body of the sprite going off the screen? Can you then amend the move **right**, **left**, **up**, and **down** blocks to include this **stay on screen** block?

Jeannette Wing, who is recognised as having defined computational thinking, refers to abstraction as the most crucial thought process in computer science. Abstractions are everywhere in computing: they hide the complexity of the underlying processes, whether that complexity is in an algorithm, a program, the computer itself, or your own understanding of what the internet is. As with all tricky concepts, the more practice and examples you engage with, the more concrete the notion becomes — even with something as abstract as abstraction. (HW)

THE PRIMM APPROACH

PRIMM is a pedagogical approach to structuring programming lessons. It follows this structure:
Predict: discuss the program and predict what it might do; what will be the output?
Run: run the program to test the predictions.
Investigate: explore the structure of the code using tracing, explaining, annotating, debugging, etc.
Modify: edit the program to change its functionality via a sequence of ever-more challenging exercises to gain confidence by extending the function of the code.
Make: design a new program that uses the same structures but solves a new problem.

You can read more at **helloworld.cc/primm**.

MATTHEW PARRY

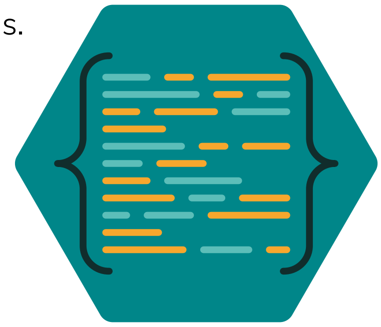
Matthew is a senior lecturer in initial teacher training at the University of Derby, UK. He is also a CAS Community Leader, a Raspberry Pi Certified Educator, a Barefoot Ambassador, and an NCCE course facilitator (**@Matthew_Parry_**).

PROGRAMMING

Programming allows learners to apply concepts from across computing in creative and innovative ways to solve problems relevant to them. Through programming, learners can create new tools and experiences, solve complex problems, and express ideas.

As learners move through school, they will progress their understanding and application of programming. Students learn to read and write simple programs from their first year of school, and over time, they develop their understanding of key programming concepts. Initially, they will focus on sequence, repetition, selection, and variables. Later, students will encounter more complex ideas, such as modularisation, recursion, and data structures. This experience usually culminates with students learning about alternative programming paradigms, including object-oriented and functional programming. Throughout this journey, students will use ever more sophisticated tools and languages.

We can apply programming across a wide range of contexts to solve a diverse range of problems. This broad application of the skill, alongside the increasing pervasiveness of computing in all areas of our lives, makes programming an important and relevant skill for all learners.



IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** programming, in summary
- **What the research says:** moving from block-based to text-based programming
- **What the research says:** levels of abstraction
- Why we should teach children to code
- Programming and the wider curriculum
- Comprehensive programming assessment

IN SUMMARY

PROGRAMMING

Read, write, test, and debug computer programs that provide meaningful output or solve a problem

STAGE 1

- Predict the outcome of a sequence of commands
- **Recognise that changing a sequence of commands can have an impact on the output**
- Create short sequences of commands for a given purpose
- **Compare programs and their output to an algorithm in order to identify issues**
- Make changes to programs when they don't behave as expected

STAGE 2

- Make predictions about the outcome of programs containing selection and repetition
- **Recognise that evaluating a condition will result in either true or false**
- Recognise that repeated sequences of commands can be replaced with a loop
- **Write programs that include selection to alter program flow based on a condition**
- Write programs that include count-controlled, condition-controlled, and infinite loops
- **Describe how variables can be used in programs**
- Describe the importance of initialisation to create consistency
- **Use tracing, testing, and debugging techniques to identify and fix issues**
- Modify or incorporate elements of existing programs to create a new program
- **Annotate code with comments to describe functionality or design decisions**

In the table below, you will find learning outcomes associated with the 'Programming' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3

- Read, trace, and predict the outcome and purpose of programs written in text- and block-based languages
- **Distinguish between event-based and procedural programming**
- Identify how subroutines can be used to make program code more manageable
- **Manipulate data held in variables, applying concatenation and arithmetic operators**
- Use lists to retrieve, add, and remove items and perform iteration on a list
- **Use a combination of logical operators to construct more complex conditions**
- Apply a systematic approach to testing programs
- **Use comments and documentation to help make programs easy to follow, test, and adapt**

STAGE 4

- **Trace and predict the outcome and purpose of programs comprising multiple functions**
- Define subroutines that make use of parameters and return values
- **Describe the properties of simple data structures such as lists and arrays**
- Implement familiar program 'patterns' to solve common problems and use appropriate libraries and modules as needed
- **Interpret and write code that contains nested statements such as nested selection and nested iteration**
- Describe how a record data structure can be implemented using dictionaries and lists
- **Manipulate variables using suitable operators or methods, converting between data types when necessary**
- Use variables within subroutines and events and distinguish between local and global variables
- **Incorporate validation techniques into programs to help minimise errors**
- Perform both iterative and final testing on programs to ensure they function correctly
- **Document a programming project to explain design decisions**

STAGE 5

- **Predict the outcome and purpose of programs written in a range of languages and using different paradigms**
- Describe the role of objects, classes, methods, and attributes within the object-oriented paradigm
- **Contrast the differences in how code is expressed in the functional programming paradigm compared to other approaches**
- Describe the role of events, messages, and the main loop in event-based programming
- **Write programs using a broad range of programming paradigms and languages**
- Write programs that use recursion to solve problems
- **Build programs that use existing or custom classes and objects**
- Create programs that implement data structures such as stacks, queues, and trees
- **Incorporate exception handling actions to make programs more robust**
- Apply a range of testing approaches to ensure functionality, performance, and robustness of programs

FROM BLOCKS TO TEXT

STORY BY Matt Hogan

Block-based programming software is an established part of any teacher's toolbox, used to introduce learners to programming. These applications have an engaging user interface with vivid coloured blocks that can be quickly snapped together and moved around on the screen to create a program. Scratch is a very popular block-based application, but there are several other similar environments, including Snap!, Pencil Code, Blockly, and more.

A regular area for debate among teachers of programming is how best to support the transition from visual programming to text-based languages. Visual languages and environments are a hugely powerful tool for introducing students to the concepts of programming, allowing them to explore concepts, solve problems, and create products through programming. However, it's important to get the experience of working in text-based languages, both to cover the curriculum and to develop the skills for the next stage. This article will explore common challenges when making the transition from blocks to text, and the tools you can use to support this.

Common challenges

Many learners have access to computers or mobile devices, or have some experience using them. Despite this, when learning to use a text-based language, there are still some major hurdles for students to overcome, many which have little to do with their ability to think logically or computationally.

Even students who have grown up with a games console, laptop, or tablet will often struggle in the early days when it comes to typing — they are often more used to typing with their thumbs on an on-screen keyboard. Young learners in particular can become very frustrated by this, as it can take several minutes to type out even the shortest of programs, and they spend a lot of their time learning the basic syntax rules, which leaves little time for them to experiment and be creative. You can use software to help correct errors in code, but these applications are often quite complicated and unsuitable for beginners.

> “ YOUNG LEARNERS OFTEN HAVE TO WASTE TIME LEARNING BASIC SYNTAX RULES, LEAVING LITTLE TIME FOR CREATIVITY

This means that learners have become accustomed to not recognising errors, such as a missing capital letter, until after they run their programs.

There can also be problems relating to a learner's literacy. Even without learning difficulties such as dyslexia and dyscalculia, many young learners are unable to recognise the difference between:

```
for i in range(5):
```

and

```
For I in ragne 5:
```

when they first type the line. Even the best programmers will occasionally make mistakes, such as typing `Flase` instead of `False`.

Rethinking the transition

Researchers Dorling and White, however, suggest that we might want to think about this transition in a different way (**helloworld.cc/dorling2015**). They explored approaches including unplugged, visual, and textual programming, and the ways students engaged with problem-solving in these contexts. They discovered that it was valuable to think about these different media as pedagogical tools, with different strengths for teaching and learning, rather than as stages students leave behind as they progress.

In their research, they outlined a pathway that used unplugged, block-based, and text-based approaches to teach the fundamental control structures of programming (sequence, repetition, function, selection, and communication). The specific languages used were Logo, Python, and Scratch; all three share similarities of sequential processing with a sprite or turtle and the reward of the instant feedback of sprite or turtle activity on the screen.

The researchers chose a cross-curricular context (geometric shapes and patterns) to teach the background of each of the control structures. By using this approach, Dorling and White concluded that "there is established pedagogy in other subjects that can be effectively used in a computing lesson. In turn, it was also evident that programming could enhance the delivery of other curricula topics."

Having a linear transition from blocks to text without any cross-curricular context may not be a suitable approach to take when learning programming. When introducing new concepts, such as local and global variables, learners may find using block-based programming

Learners don't necessarily need to leave block-based programming behind as they progress

languages easier to use and understand. It gives learners an opportunity to see the similarities between two languages and a clear context that they understand.

Tools to support

When learners are new to text-based programming, they may feel apprehensive about learning code. It is worth discussing how they are feeling and trying to mitigate any risk of them being put off text-based programming. Hybrid platforms such as the micro:bit MakeCode editor and EduBlocks are a great way of doing this. These online platforms have developed a block-to-text feature, which includes a tool to drag and drop blocks of code, similar to Scratch, although the blocks correspond more directly to Python code. These online platforms also give the ability to toggle between blocks and Python code, or to split the screen to view blocks and code next to each other.

These tools help to bridge the gap between block-based and text-based programming languages by:

- Using a familiar drag-and-drop user interface
- Allowing users to transition between block-based and text-based methods while creating a program
- Helping learners become familiar with Python code in a way that reduces errors that could arise when typing code into the micro:bit editor

Dual-modality environments

Some applications, for example Pencil Code, offer a dual-modality (or hybrid) environment, in which learners can write and see the same program in a text-based and a block-based programming environment side by side. The key point of dual-modality environments is that the code is displayed automatically in the alternative text-based format: block-based code is automatically translated to text-based code, and vice versa. In a 2017 study, researcher David Weintrop set out to test the assumption at the heart of dual-modality applications, that being able to match a text-based program to its block-based equivalent supports the development of understanding program syntax in a text-based language (**helloworld.cc/weintrop2018**).

Weintrop carried out a 15-week study with 14-to-16-year-old students in the US to investigate the differences in learning when using block-based, text-based, or hybrid (a mixture of both, using a dual-modality platform) programming tools. The 90 students in the study were divided into three groups. Each group was set the same tasks with the same learning objectives, but they used either block-based programming, text-based programming, or the hybrid environment. After five weeks, students were given a test to assess learning outcomes, and they were asked questions about their attitudes to programming, specifically their perception of computing and their cofidence with programming. After another ten weeks, all the students were taught Java, and then the test and attitudinal questions were taken again.

The results showed that at the five-week point, the students who had used block-based programming scored more highly in their learning outcome assessment than the text and hybrid groups, but at the final assessment after 15 weeks, the scores of all the groups were roughly equivalent. In terms of students' perceptions of computing and their confidence levels, the responses of the block-based group were very positive at the five-week point, but less positive after 15 weeks. Taking both methods of assessment into account, the hybrid group showed the best results, and the use of block-based programming did not hamper students' transition to text-based programming. Although more research is needed to support the conclusions of Weintrop's study, the approach of the hybrid group can be adopted in many classrooms.

Block-based languages are valuable when introducing any new programming concept to learners, and the development of more and more hybrid models makes it easier for learners to identify the similarities between blocks and text. Perhaps there is no linear path from blocks to text; maybe the important thing is that teachers know and understand the tools that can be added to their toolbox in order for them to teach programming effectively to their own learners, in their own specific context. (HW)

FURTHER READING

- ✓ Dorling, M., & White, D. (2015). Scratch: A Way to Logo and Python. *SIGCSE '15, March 04 -07 2015*, Kansas City, USA. **helloworld.cc/dorling2015**
- ✓ Weintrop, D., & Wilensky, U. (2018). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction.* **helloworld.cc/weintrop2018**

LEVELS OF ABSTRACTION

STORY BY James Robinson, Andy Bush, and Sway Grantham

Learning to program is broader than simply learning to write code. While coding is a big part of the programming process, programming also encompasses analysing and understanding the task or problem being addressed, designing a solution, and testing and debugging the program. If we try to teach all these things to novice programmers at the same time, the cognitive load overwhelms them. The skill of abstraction — routinely adjusting your focus and the level you're working at while developing a programmed solution — is therefore a vital skill to introduce to learners. These different perspectives or levels can be modelled by the levels of abstraction (LOA) hierarchy. There are several variations of this hierarchy, but this article discusses the model outlined by researcher Jane Waite and her colleagues (helloworld.cc/waite18).

The LOA framework

This hierarchy emphasises the critical role that abstraction plays in developing programs. It describes four levels, encompassing different degrees of abstraction. We can characterise the four levels as follows:

TASK

The task outlines the problem to be solved, or describes what the project should actually do.

With younger learners, a teacher often defines a task. As students become more experienced, they can expand a given task or develop a task themselves. Later, they may work from formal specifications or user requirements, and may even define the task independently through user research.

DESIGN

The design level includes the algorithm, which outlines the process and logic that will exist within the program. The design may also contain other aspects, such as artwork, sounds, and sketches of what the project will look like, or how it will be put together. This level contains more detail than the overall task, but doesn't yet refer to the code or programming languages that will be used. Learners can use a range of tools to represent their design, including text, sketches, flowcharts, and diagrams.

CODE/BUILD

The code level (or build level, for physical computing projects) represents a static program that implements the design from the level above. This could be constructed in any number of programming languages, including both block- and text-based programs. Learners will be limited to the languages and tools they are familiar with initially. As their confidence and repertoire increases, they can be more discerning about choosing the best tool to implement their design.

RUN THE PROJECT

At the lowest level, the programmer is concerned with how the program behaves when it is run. Does it run? Are there errors? Does the program behave as expected? These are all important questions at this level. Here, learners need to know how to test their programs, find and correct errors, and trace the execution of their code to ensure they understand its behaviour.

These levels of abstraction do not represent a linear pathway to developing a project. While learners will often begin at the task level and will generally progress towards running the project, they will frequently need to switch back and forth between levels.

LOA in practice

While expert programmers regularly move through these different levels without even realising it, it's important to scaffold them for novice learners. This means that when you are planning a programming unit of work, you need to consider which level the lesson is working at, and what support learners might need at each level. You should use this approach across different projects, genres of program, and age groups. This will ensure that as learners begin to move from novice to expert, progressing through each of the levels becomes routine. The rest of this article will now exemplify this through a programming project from the Teach Computing Curriculum for learners aged ten to eleven (**helloworld.cc/tccLOA**). Learners should complete the project mostly independently so you can assess their understanding of variables across three one-hour lessons.

THE TASK

Give students the following task:

Create a 'catching' game that includes a score and at least three falling objects. The objects should fall at different speeds.

As a teacher, I want the learners to demonstrate their understanding of variables. This means I don't want them to choose a task that might limit this, and so in this project, I have provided the task. In another unit, the focus will be different and I might work with learners to identify the user requirements themselves.

THE DESIGN

For context, these learners will have had several years' experience working through the levels of abstraction, so they know how to use designs and implement them. Before this task, they will also have completed three lessons introducing the concept of variables.

Every programming language is slightly different in how it can be used. It is important for learners to have an understanding of this before they begin designing, or they may plan a project that is unachievable. In this project, we provide a 'program stub', which is part of the program they will need to complete the task (**helloworld.cc/fruitcatcher**). Give learners time to familiarise themselves with this project before they begin their design, so they can remind themselves of the Scratch environment and its capabilities. The stub will also maintain learners' focus on variables. Providing the code for the sprites that will not use variables means learners will spend their time designing and implementing the parts of the program that do require an understanding of variables. This also means that the activities will largely be at the designing and coding levels of abstraction.

The main design activities can now begin. Learners need to choose their assets (sprites, sounds, and backgrounds). This is also their opportunity to begin personalising their project; it's important that learners recognise there is no one right answer, and that they can achieve the task in different ways.

Just like the program stub we provided, the algorithm and design for that part of the program have been provided for learners (**Figure 1**). This models to learners how we write algorithms, and create this part of the program design. It also gives them a reference for their next step:

Write two algorithms (include a drawing and a description) for the two new sprites you chose. Each sprite must change the score by a different amount, should move at a different speed, and could be a different size.

Again, you can see how this part of the task guides the design to include a focus on the variables (score, speed, and size), but leaves learners free to design their own programs independently. The last step, before we move on to the coding level, is to encourage students to check back with their original task. Does their design meet the requirements you gave them? Once they are confident that it does, they can begin their implementation at the coding level.

> AS LEARNERS MOVE FROM NOVICE TO EXPERT, PROGRESSING THROUGH EACH OF THESE LEVELS SHOULD BECOME ROUTINE

THE CODING/BUILDING

Learners can refer to **Figure 1** and its implementation in the program stub to scaffold what they need to do next. The unit of work also contains a design worksheet, which is organised so that learners can approach each section independently:

1. Add sprites with the necessary artwork
2. Initialise variables — use clear names!
3. Algorithm for sprite 1
4. Algorithm for sprite 2

Algorithm for the falling star

The falling star moves down by eight from a random x position at the top of the screen.

If the sprite falls onto the bowl, change the score by three. It falls again from a random x position at the top of the screen.

If the falling star touches the screen bottom, it falls again from a random x position at the top of the screen.

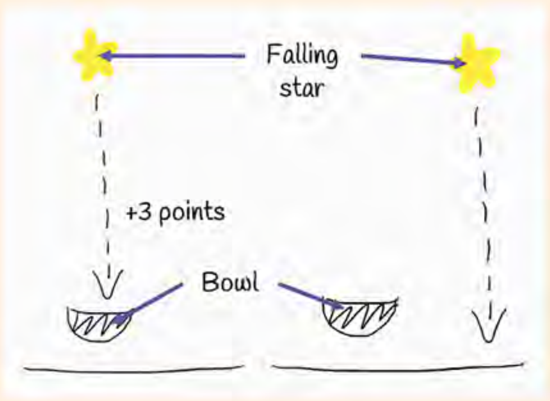


Figure 1 The information above prompts learners how to write the algorithm and create the program design for this task

> IF LEARNERS START CHANGING RANDOM VALUES 'JUST IN CASE' , YOU'LL KNOW IT'S TIME TO BRING THEM BACK A LEVEL

As this project is an assessment piece, learners are largely left to implement their programs with this scaffolding. If learners do not have much experience at implementing code, you might spend more time making links between the algorithm and the code.

Working at the code level is also an opportunity to observe more general programming skills: are learners testing each bit of code as they go? Do they duplicate chunks of code rather than rewriting very similar scripts for each sprite? These will be opportunities for you to engage with learners while they are creating their projects, depending on what the focus of your unit is.

RUNNING THE PROJECT

It can sometimes be difficult to distinguish between the levels of coding/building and running the project, as learners should be repeatedly cycling through coding and running, to check t hat their program is working. This can become a problem if learners find a bug in their programming. Most debugging will happen at the coding level, especially in the early stage of text-based languages where many of the errors are syntax-related. However, if learners find themselves with a logic error, or a more substantial bug, they may need to revisit their algorithm to work out what it is doing and to plan an alternative solution. An easy way to spot whether you need to bring learners back to the design level is if their approach to debugging has resorted to changing random blocks/values 'just in case' because they have no idea why their program is not working. Talking through the algorithm and precisely what it is doing, and then working out the bit that's not clear and rewriting that section, will support these learners.

The other aspect of this level that should not be overlooked is testing. This is not the same as just running the code as you are working through the project. Testing is about checking whether your program behaves as expected for the task or design. In this project, for example, a program might run successfully, but perhaps the items of fruit are flying up into the air instead of into the fruit-catcher bowl. This is what testing is for.

Working through these levels within a unit of work gives learners the opportunity to develop skills at a realistic pace and gives you the opportunity to scaffold specific skill sets. This ultimately ensures that learning to program is an option for a wider range of learners. (HW)

FURTHER READING

✓ The National Centre for Computing Education. (2022). *Programming and Algorithms within the Computing Curriculum.* **helloworld.cc/p&areport**

✓ Waite, J. L, Curzon, P., Marsh, W., Sentance, S., & Hadwen-Bennett, A. (2018). Abstraction in action: K-5 teachers' uses of levels of abstraction, particularly the design level, in teaching programming. *International Journal of Computer Science Education in Schools, 2*(1), 14-40. **helloworld.cc/waite18**

WHY WE SHOULD TEACH CHILDREN TO CODE

Simon Peyton Jones explores how programming can bring computer science to life for learners

In a March 2019 blog post (helloworld.cc/schleicher2019), Andreas Schleicher, director of education and skills at the OECD, asked, "Should schools teach coding?" This was somewhat misreported in the press as "Teaching children coding is a waste of time, OECD chief says" (helloworld.cc/telegraph2019). But it's a good question.

Let's start at the beginning, though. Technology moves fast. To equip our young people to flourish in a world of change, we therefore strive to give them a foundational understanding of the world that surrounds them, and an intellectual toolbox that will equip them to deal with successive waves of technology. For that reason, the computing curriculum in England, introduced in September 2014, established computer science (not just coding, and with computational thinking at its core) as a foundational subject that all children learn, alongside maths and science, from primary school onwards. The previous ICT curriculum focused on technology; the current curriculum focuses on ideas and principles. As the famous aphorism puts it, "Computer science is no more about computers than astronomy is about telescopes."

SIMON PEYTON JONES
Simon is a British computer scientist who researches the implementation and applications of functional programming languages, particularly lazy functional programming.

But if computer science is not about computers, what is it about? It is the study of information, computation, and communication. Take information, for example: suppose I show you a picture of the French national flag, and one of the *Mona Lisa*, and ask, 'Which picture contains more information?' What does that even mean? A way to make the question more precise might be, 'Suppose I dictated instructions to you over the phone; which picture would take you longer to reproduce?'

Clearly the *Mona Lisa* has more information in this sense: it would be slow and painstaking for me to dictate instructions so that you could reproduce it at your end, even rather approximately. Thus, we have begun to speak of information as a measurable quantity. We start to think about how tightly we could compress data before transmitting it, and how we could detect, and perhaps correct, errors made during transmission. All this is called information theory; it is part of computer science, it has a substantial body of theory, and it has immediate practical consequences.

Where coding fits in

What, then, is the role of coding or programming (the terms are roughly equivalent) in computer science? Coding is not the message of England's computing curriculum, but its medium. Coding is the lab work of computer science: it motivates, illuminates, and brings to life the dry bones of theory. Without programming, computer science would be a dry, theoretical husk of a subject. Imagine a music lesson in which the students only studied the rules of counterpoint or the structure of a sonata, but never brought them to life by performing or composing any music.

But that's not all: programming is more than mere medium. As Fred Brooks put it, "The scientist builds in order to study, but the engineer studies in order to build." Most of programming's body of knowledge is organised around the challenge of building ever more ambitious edifices of software, and having them actually work and be useful. Programming is the very stuff of computer science.

Coding is phenomenally creative. The same Fred Brooks wrote that, "The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realising grand conceptual structures."

When a child does a science experiment, she is seeing physical principles at work, coming to life in front of her eyes. If she does the experiment right, we know what will happen. In contrast, when she writes a program, no one knows what will happen. The programmer brings into the world a

FEATURE

Without programming, computer science would be a dry, theoretical husk of a subject

new creation, formed from an infinitely malleable substance, which does something new, conjured from the mind of its creator. We are not limited by the strength of wood, or the budget of the school workshop: in programming, we are limited only by our own ability or inability to manage the complexity of our creation.

Self-assessment and risks

Coding offers immediate, tangible feedback. No need to wait for the teacher to mark your essay, as in English; in computing, the program runs, or not, and remorselessly exposes the logical errors in your thinking. When hunting a bug in a malfunctioning program, we form a hypothesis about what is wrong. We formulate tests that will confirm or refute that hypothesis. In the light of the results of those tests, we refine the hypothesis, and so on; it is the scientific method in action. Even for students who will never explicitly program again, programming teaches understanding and reasoning skills that are needed by everyone: business innovators (for identifying a need or potential); scientists (for working with data and developing computational models of scientific processes); those procuring software (for example in health services, for knowing what is possible and what they should be looking for); or end users (because one must always have a notional machine model of what a given piece of software is doing).

Programming is a tremendously useful and marketable skill. In every corner of business, and in every part of our daily lives, there are programs, and they all need to be written, modified, fixed, and stitched together. There is tremendous demand for skilled programmers, who command high salaries as a result.

And yet, and yet. There are two risks here. First, there is the risk that we confuse the medium with the message. I fear a future prime minister giving a speech saying, "The new computing curriculum has been a great success: every child now leaves school fluent in Python." What a disaster that would be! The computing curriculum is focused on ideas and principles, not on a particular technology such as Python. Yes, some of those ideas (sequence, iteration, choice, abstraction) are directly embodied and brought to life in Python, but Python is just one embodiment among many, not the thing itself. Once pupils have learnt to code in one language, they should be able to

WE ARE LIMITED ONLY BY OUR OWN ABILITY OR INABILITY TO MANAGE THE COMPLEXITY OF OUR CREATION

Whatever learners want to be when they grow up, learning to program teaches them vital skills

"ONCE PUPILS HAVE LEARNT TO CODE IN ONE LANGUAGE, THEY SHOULD BE ABLE TO QUICKLY TEACH THEMSELVES OTHERS

quickly teach themselves others built on the same concepts, and should also be able to recognise those same concepts appearing in the wider world that surrounds them.

The second risk is that we may forget that the school computing curriculum is for the many, not the few. I certainly hope that the education our young people receive will inspire some of them to be the software developers of the future. But many more will become lawyers and plumbers, hairdressers and doctors. They all learn the elementary principles of natural science, and similarly they should all learn the elementary principles of computer science. And, just as mathematics appears in primary schools mainly in the guise of arithmetic, so computer science will appear mainly in the form of simple programming. Just as no one confuses arithmetic with the manifold glories of mathematics, so we should not confuse programming with computer science.

Tomorrow's problems

Returning to Schleicher's blog post, he says, "The risk is that we will again be teaching students today's techniques to solve tomorrow's problems; by the time today's students graduate, these techniques might already be obsolete. We should instead focus on the computational thinking that underpins these techniques, and that students can use to shape the technologies of tomorrow."

Fair enough — and indeed, computational thinking is already explicitly at the core of the English national curriculum, from start to finish. But teaching programming is emphatically not 'teaching today's techniques to solve tomorrow's problems'. Programming is computational thinking incarnate, brought to life, made tangible, executable, and useful. It provides a powerful way to practise and so develop those computational thinking skills, and understand them deeply. People occasionally say, 'In the future, computers will program themselves,' but I believe they are mistaken — we will simply increase the ambition of the programs we write.

So yes, to answer the question, we should teach our children to code. But we should do so not as an end in itself, but rather as a powerful and effective means to motivate, illuminate, and exemplify the underlying principles of computer science. There is no more intellectually exciting, creative, or practically useful subject. I want to convey to our young people a visceral sense of that richness and creative possibility, and by far the best way to do so is to share with them the joy and beauty of programming. (HW)

FURTHER READING

Andreas Schleicher's blog post: **helloworld.cc/schleicher2019**

The Telegraph's interpretation of Schleicher's blog post: **helloworld.cc/telegraph2019**

INTRODUCING PROGRAMMING THROUGH THE WIDER CURRICULUM

Ben Hall looks at the crossover between literacy and coding, and how it can help in the primary classroom

Much of the research around how we learn to program, and which approaches are most successful, has been focused on older learners, particularly undergraduates or those transitioning to text-based languages. There is comparatively little research on how children learn to program from an early age. If we want children to become curious and confident programmers, then an understanding of their formative experience would certainly help.

Computing is usually seen as a STEAM (science, technology, engineering, arts, and maths) subject. However, are we missing a trick? Are there parallels between programming and literacy, and could this broaden the appeal of the subject for both learners and teachers?

Researcher Jane Waite and her colleagues drew parallels between the design level of a program — where a pupil uses simple language to explain what characters will be needed and what they will do — and the planning of writing in a literacy lesson (**helloworld.cc/waite2018**). Researcher Raymond Lister and colleagues also investigated the links between writing skills and early programming, but in the context of text-based (in this case Python) languages for older learners (**helloworld.cc/lister2009**). Can this research help us develop our understanding of how the youngest learners are introduced to programming?

Skill acquisition

As a former specialist computing teacher in a primary school, I have been fortunate to teach learners from the age of six right through to the age of thirteen. This has given me some insight into how children pick up key skills.

I've typically introduced computing to the youngest learners by looking at concepts such as instructions, and developing vocabulary to help learners access floor robots such as Bee-Bots (see the photo below) or Code-a-Pillars. This works really well in a continuous provision environment where children immerse themselves in the language before applying it to a different context, much as you might do with many other subjects. Using Fakebots or grids can be a really useful way to introduce directional language (see the article on page 60 for more on this). Once the language is secure, we can then introduce symbols to represent movements.

This is very similar to literacy, where children are immersed in the language through opportunities to listen to stories and explore books. Through this, they understand that books convey meaning and that they have an order and a sequence: spot the link? Sway Grantham, a senior learning manager at the Raspberry Pi Foundation, explores this connection in a blog post, analysing 'Talk for Coding' as an approach based on Pie Corbett's 'Talk for Writing' learning sequence (**helloworld.cc/grantham2017**). She draws upon research that concluded that sequence, structure, and clarity of expression are as important in programming as they are in writing (**helloworld.cc/burke2010**).

Parallels

As children's literacy skills develop and they become more aware of structure in writing, there are some interesting parallels we can draw on.

Firstly, nursery rhymes. They're sequenced (for example, *One, Two, Three, Four, Five*), and may include some form of repetition (such as *Hickory Dickory Dock* or *Ten Green Bottles*). Knowledge of these patterns can be directly applied to a sequence of instructions, or even used as a basis for programming projects. In ScratchJr, we have the perfect platform for storytelling. One of my favourite learning sequences with Year 1 (aged 5–6) tied in with their class book at the time, *The Three Little Pigs*, which they retold through ScratchJr.

I was careful not to bring in the programming element too quickly; it was much more effective when children were able to plan an element, using a storyboard, which they could then apply within the tool. This is very similar to how most literacy teachers would encourage children to plan a story before writing it. Design in programming is equally important, and this is a way to introduce it by using a known context and drawing from prior learning. There are many parallels with literacy — we plan writing and design algorithms, and storyboards can be used for both.

The research suggests that this approach is not widely used: Waite and colleagues identified that 82 percent of primary teachers thought that design in programming was at least very useful, whereas only 44 percent actually used it in their teaching 'always' or 'usually'. Double this number 'always' or 'usually' used planning in writing. This could reflect a lack of confidence in teachers' subject knowledge, a shortage of materials to support design-led activities, or the curriculum itself. In England, the curriculum for English states that children should 'plan, evaluate, and improve' their writing — but there is no mention of planning or design in computing.

It can be helpful to draw upon other subjects, not just literacy, in your primary computing classes. Consider languages. How could learners apply their experience of learning a second spoken language to that of learning a programming language? In maths, when you teach patterns and sequencing, how can you translate these into the corresponding programming concepts? Can you link debugging to the process of correcting errors in other subjects? A great way to develop your computing teaching is to ensure you use all your experience, from all of your primary curriculum, to make computing as accessible as possible to the broadest range of learners. (HW)

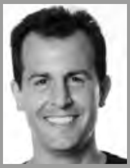
SEQUENCE, STRUCTURE, AND CLARITY OF EXPRESSION ARE AS IMPORTANT IN PROGRAMMING AS THEY ARE IN WRITING

BEN HALL
Ben is a learning manager at the Raspberry Pi Foundation, developing curriculum resources for England's National Centre for Computing Education. He was previously a primary teacher specialising in computing (**@hengehall**).

FEATURE

COMPREHENSIVE PROGRAMMING ASSESSMENT

Establishing whether a program works needn't be the only goal of assessment



DAVID J. MALAN

David is a Gordon McKay Professor of the Practice of Computer Science at the Harvard John A. Paulson School of Engineering and Applied Sciences in Massachusetts, USA. He is the instructor for CS50.

DOUG LLOYD

Doug is senior preceptor in computer science at the Harvard University Division of Continuing Education in Massachusetts, USA. He is the course manager for CS50.

Determining whether a student's code is correct is an important part of the grading process. Whether you use an autograder, or benchmark student code through a series of manual unit tests, it's fairly easy to determine whether a student has considered the solutions to your test cases in their solution. But while correctness is important, there are other important questions to consider. For example:

- Did the student at least attempt to solve the program at hand, even if they struggled along the way with syntax, such that their code might not compile? Particularly among students less comfortable with coding, rewarding effort is a reminder that it's OK to struggle.
- Is it actually possible to read the student's code? Poor indentation (unless programming in Python!) and no commenting might not matter to the processor, but to an instructor it can cause quite the headache!
- How efficient is the student's code? If the program is correct, but takes 15 minutes to run, is it all that useful?

At CS50, Harvard University and Yale University's largest open-learning course, we respectively grade for scope, style, and design, in addition to assessing for correctness. It is design, however, that we'll discuss further here.

What to look for

If correctness asks the question 'Does it work?', then design asks the question 'How does it work?' Our goal in assessing design is to get students thinking critically about the decisions they make in their code, and to put them into a mindset of continuous improvement. We rarely give out a perfect score in design, as there are almost always areas where we can make improvements. We ourselves are in a mindset of continuous improvement, and we feel that if we can find opportunities to design better solutions to our problem sets (and year after year, we almost always do), our students should be able to as well.

CS50 staff grading student work at a grading party

Students get help from *CS50* staff members during office hours

Some questions that we consider when looking at design include:

- How frequently do the same lines of code repeat?
- Does a student's code have an over-reliance on loops or magic numbers?
- Did the student choose the most efficient algorithm?
- Was the code broken into functions or subroutines where appropriate?
- How many lines of code did the student write? The length of a student's submission is often, though not always, a good indicator of how well-designed that code is; if fewer lines of code are needed, the student is probably taking advantage of an efficiency.

A MULTI-AXIS GRADING PHILOSOPHY

Assessing programming through different axes allows us to test several skills via a single assignment:

- **Scope** rewards student effort in solving problems, reiterating the importance of a culture of error and normalising the idea that failure to completely solve a problem is not a total failure, but a learning opportunity (did you try?)
- **Correctness** judges the performance of a student's code, either using a test harness or via manual execution of the code (does it work?)
- **Design** considers the efficiency, elegance, and clarity of a student's code from an organisational standpoint (how does it work?)
- **Style** is the most human-focused axis, and considers how readable and well-commented the code is for others who might need to read it (how does it look?)

Trade-offs

Unlike the other three axes — correctness, which we can assess with autograders; style, which we can assess with linters; and scope, which a quick glance can assess — grading design can take significant time and effort for experienced and inexperienced teachers alike. Students in CS50 submit their code to course staff via GitHub, and teaching fellows can offer feedback via comments on GitHub's web interface. The ability to use 'saved replies' for issues that might recur in the work of multiple students can save some time over writing the same comment out again, and grading design only becomes quicker and easier once you've seen multiple different solutions to the same problem, and you're familiar with common errors.

That said, the qualitative feedback received via the design axis is, in our opinion, the most valuable. Comments on design should get students thinking about what makes one solution better (or worse) than another, even if both have the same output. This reinforces the notion that there is not always one right answer, and we hope it will encourage students to be more aware of their programming decision-making. (HW)

DATA AND INFORMATION

Data and information is a strand of computing that focuses on how data is stored, organised, and used to represent the real world and provide meaningful insights. Most data collection, storage, and analysis will involve computers, and as such, this area of computing particularly complements other school subjects, with a practical focus on collecting, storing, and analysing data at scale. The study of data and information also provides a solid basis for learners to understand the role of artificial intelligence in analysing and interpreting data.

There are skills and concepts associated with each stage of a data life cycle, which involves questioning, collecting, implementing, analysing, and sharing. The implementation stage of this process, in particular, involves the application of computing. Learners will use, design, and compare different tools and approaches to storing data digitally, including text files, spreadsheets, and databases.

IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** data and information, in summary
- **What the research says:** fostering data literacy competencies
- Captivating data visualisations
- A real-life data project
- Boosting engagement with big data

IN SUMMARY

DATA AND INFORMATION

Understand how data is collected, organised, and analysed to explore real-world scenarios

STAGE 1

- Explain that data can be helpful in answering questions
- **Collect, group, and compare simple data**
- Explain that names/labels can be used to describe objects
- **Identify how attributes can be used to compare objects**
- Describe how data can be presented pictographically
- **Present data using pictographs and simple charts**

STAGE 2

- Identify that yes/no questions can be used to structure data
- **Answer questions using decision trees**
- Use sensors/data loggers to collect data
- **Explain how data can be searched and ordered using different fields**
- Use suitable tools and applications (including spreadsheets) to handle data
- **Use AND/OR operators to refine searches**
- Choose an appropriate format to present data

In the table below, you will find learning outcomes associated with the 'Data and information' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Identify the differences between data and information	Design a model to represent specific objects and relationships within a scenario	Produce a relational data model to represent a scenario
Describe the stages of the data life cycle, including questioning, collecting, implementing, analysing, and sharing	**Describe the role of a database and its data model in structuring and organising data**	**Describe approaches to optimising databases, including normalisation**
Collect, process, and analyse quantitative data to answer a specific question	Explain how a relational database is implemented through tables and keys	Explain how databases handle and process multiple transactions
Explain the need for being selective when working with large data sets	**Compose SQL expressions that add, retrieve, and update data in an existing database**	**Construct database tables, relationships, and views using SQL**
Apply formulas within a spreadsheet to process data	Explain how data can be organised, stored, and retrieved using files	Design, build, and interact with a database using a database management system
Use sorting and filtering techniques to help identify patterns in data	**Identify sources of potential bias within a data set at each part of the data life cycle**	**Describe the relationship between artificial intelligence, machine learning, and big data**
Present data in a range of different formats to aid understanding	Distinguish between quantitative and qualitative data	Describe the kinds of problem that data science can address
	Prepare data for analysis, importing, structuring, and cleaning as appropriate	**Design and conduct a data science investigation, identifying a problem, data sources, and methodology**
	Report on findings, making use of descriptive statistics and visualisations	Build interactive data dashboards to summarise data and aid analyses
	Distinguish between public and private data sets and work with both	**Identify potential sources of bias within data analysis**

FOSTERING DATA LITERACY COMPETENCIES IN SCHOOL

STORY BY Andreas Grillenberger

Today, data analysis is everywhere: large companies collect and analyse masses of data to systematically promote their products, while social media platforms use data analysis to suggest friends.

Nowadays, everyone is confronted with new challenges because of the increasing and widespread relevance of data; these include deciding which personal and foreign data is shared with others (including services on the internet), under which conditions, and for which purpose. We also need to ask what others can do with, and read from, this data. And it's not just children and young people who need to gain skills in this area — we all need to become data literate.

Data literacy competencies

But what does data literacy mean? According to widely accepted definitions, data-literate people are able to work with and handle data in a meaningful way — for example, by acquiring, structuring, or analysing it.

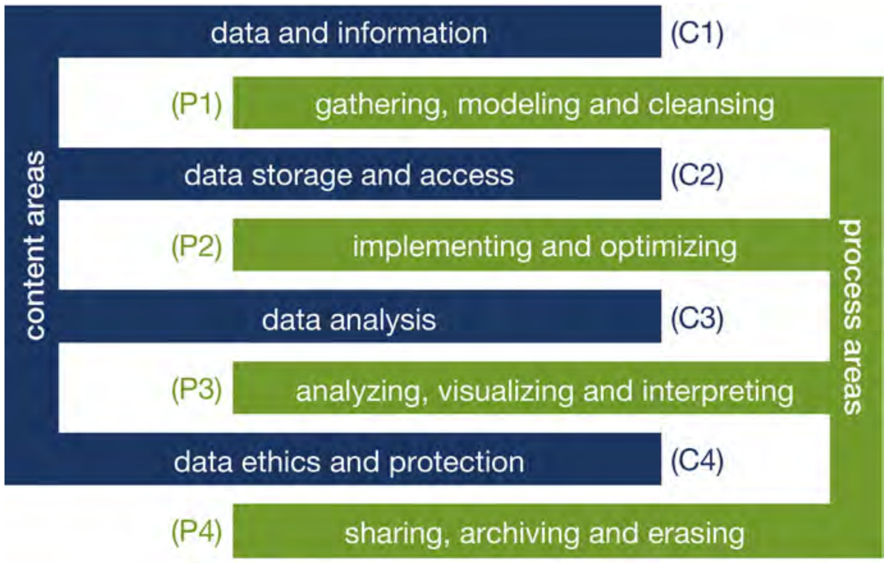
In recent years, I have investigated this topic further with the Friedrich-Alexander-Universität Erlangen-Nürnberg Computing Education Group. We have looked at data literacy from a computing education point of view, taking into account various perspectives. We looked at the technical perspective on the large topic of data, and we also considered students' and teachers' perspectives, as well as requirements from society. On this basis, we developed a data literacy competency model (**Figure 1**).

This model characterises data-related competencies from two different perspectives: content areas and process areas. The content areas clearly emphasise technical aspects and hence are focused on computer science content. In contrast, the process areas take a more practically oriented perspective, illustrating what can be done with data.

These two types of competency area are closely intertwined, meaning that each data literacy competency has to connect to at least one content and one process area. For example, the competency to visualise data and analyse results incorporates both a content aspect (such as knowing different visualisation methods and their purpose) and a process aspect (such as being able to prepare data in a way that is suitable for visualising it and creating the aspired visualisation). Although the competency model was developed with a focus on computing education, we can also adapt it to incorporate aspects from other subjects. After all, computer science is not the only subject that is dealing with data today. Other topics can contribute important elements too, particularly to the content areas, enriching the model and extending its usability.

The life cycle of data

When trying to include data literacy competencies in school teaching, the question often arises as to where to start.



content areas
data and information (C1)
(P1) gathering, modeling and cleansing
data storage and access (C2)
(P2) implementing and optimizing
data analysis (C3)
(P3) analyzing, visualizing and interpreting
data ethics and protection (C4)
(P4) sharing, archiving and erasing
process areas

Figure 1 According to the research-based data literacy competency model, each data literacy competency must have both content-related and process-related aspects

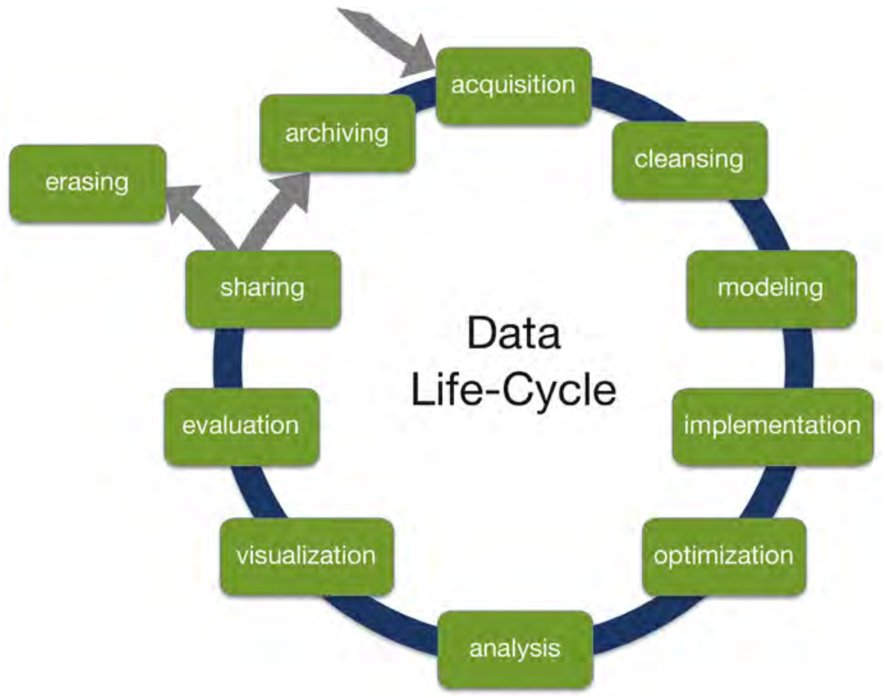


Figure 2 The data life cycle model gives a structure for teaching data literacy, for both teachers and students

FURTHER READING

✓ Grillenberger, A. & Romeike, R. (2018). Developing a Theoretically Founded Data Literacy Competency Model *WiPSCE '18, October 4-6, 2018,* Potsdam, Germany. **helloworld.cc/grillenberger2018**

Most computing lesson plans probably have various connection points to data literacy, so there are multiple possibilities for data literacy teaching. Yet it is crucial to keep in mind the entire process of working with data. When only discussing distinct parts of this topic (for example the analysis), other important aspects are missing (such as gathering data, or justifying the analysis from an ethical perspective).

We therefore also developed the data life cycle model (**Figure 2**) as a structure for teaching data literacy. This model provides a structure for teachers and students when working with data and emphasises the complete process, not just a small part of it. Of course, we can't consider all aspects in the same depth in school, but using the data life cycle helps to bring together all the knowledge and skills students gain throughout their computing education.

For example, aspects related to data modelling, implementation, and optimisation are already in most computing curricula, and so only need to be connected with other elements of the data life cycle. When working with databases, you can gather and structure real data in class for efficiency, storing them in the database instead of discussing fictitious examples. The important question, therefore, is not where to start with teaching data literacy, but where to connect it to what we already teach. The data life cycle helps to identify such connection points.

Fostering data literacy in school

Several challenges have to be overcome when fostering data literacy competencies in school. For example, we have to identify suitable tools, select appropriate examples that are relevant and motivating to students, and work out which concepts can foster these skills.

We cannot teach most data literacy competencies through theory alone; suitable examples and appropriate data play a significant role in teaching these competencies. Such data is available from various sources today. The application programming interfaces (APIs) of widely known services on the internet (such as Twitter) aren't the only data sources we can use; there are also rich and easy-to-use data sets available, such as those released by public administrations as part of open-data projects. For example, **data.gov.uk** contains open data published by the UK government and public bodies.

An exemplary project, particularly considering data analysis, is based on real data about school students. For example, learners could examine a data set about Portuguese students that was released on the UCI Machine Learning Repository, **helloworld.cc/UCIdataset**. Students can analyse this using simple tools (like Orange, **orange.biolab.si**), to predict grades based on the information contained in the data set.

As this setting directly concerns students, particularly if the teacher presents it as a possible new way to grade them, it raises several ethical problems that directly affect the students. The resulting discussions that arise on challenges, risks, and opportunities, along with the possibility of directly working with and analysing large amounts of data, lead to another challenge. Although this article has taken a computing education perspective on data literacy, the topic also affects and applies to many other subjects. We should therefore consider data literacy an interdisciplinary topic, and it should be taught in school accordingly.

Fostering data literacy in school is an open challenge to which we all can, and must, contribute if we are to prepare our students for a life in a world where we use data continuously and everywhere. (HW)

DATA VISUALISATIONS FOR INQUISITIVE MINDS

How to get your students started with data science by introducing exciting and interesting visualisations

Visualising data is the art of being able not only to present data in a visual format, but to truly tell a story with it. A good visualisation will bring data to life, aiming to provide the audience with something that they might not have been able to spot merely by looking at numbers on a page. While writing a unit on data science for the Teach Computing Curriculum, I became infatuated with the creative ways in which people have visualised data, drawing out trends, correlations, and patterns. In this article, I want to highlight some of the best visualisations that are adaptable for most age groups, so you can use them to inspire your classes and get them thinking about data.

Telling a story with data

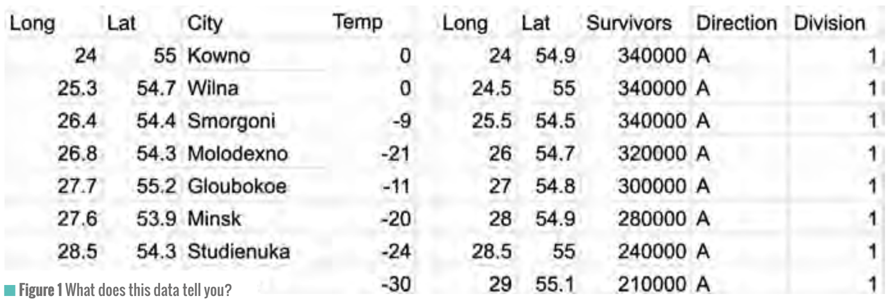
Take a look at the data you can see in **Figure 1** below. Can you extract any meaning from it? What story is it telling you? You can inspect the data in more detail here: **ncce.io/minard-data**.

This data relates to Napoleon's 1812 march on Russia. The numbers alone don't tell much of a story, but in 1869 Charles Joseph Minard, a French civil engineer known for his information graphics, produced what is now widely regarded as the best statistical graph of all time; see **Figure 2**.

The visualisation shows Napoleon's army, departing at full strength from the Polish border to Russia, and then their subsequent retreat. The thickness of the lines represents the size of the army, with the beige line representing the march on Moscow and the black line representing the retreat back to Poland. The scale of the losses becomes very clear when you compare the thickness of the two lines: Napoleon set off with 422,000 troops and returned with just 10,000.

Other forms of data have also been represented in the visualisation. Geographical features such as locations and rivers crossed have been plotted, as well as the varying temperatures at different points on the march and retreat. If you look closely, this helps you to see where and why tragedy occurred at various points on the journey. During the retreat on 28 September 1813, the data labels on the black line read 50,000, before dropping to 28,000, showing that 22,000 men died crossing the Berezina river near Minsk!

What I love about this visualisation is that the more you look at it, the more you see. The visualisation really made me feel as though I understood a little more about the story of Napoleon's march on Russia — a story that I certainly wouldn't have been able to get by simply studying the data as numbers on a page.



Long	Lat	City	Temp	Long	Lat	Survivors	Direction	Division
24	55	Kowno	0	24	54.9	340000	A	1
25.3	54.7	Wilna	0	24.5	55	340000	A	1
26.4	54.4	Smorgoni	-9	25.5	54.5	340000	A	1
26.8	54.3	Molodexno	-21	26	54.7	320000	A	1
27.7	55.2	Gloubokoe	-11	27	54.8	300000	A	1
27.6	53.9	Minsk	-20	28	54.9	280000	A	1
28.5	54.3	Studienuka	-24	28.5	55	240000	A	1
			-30	29	55.1	210000	A	1

Figure 1 What does this data tell you?

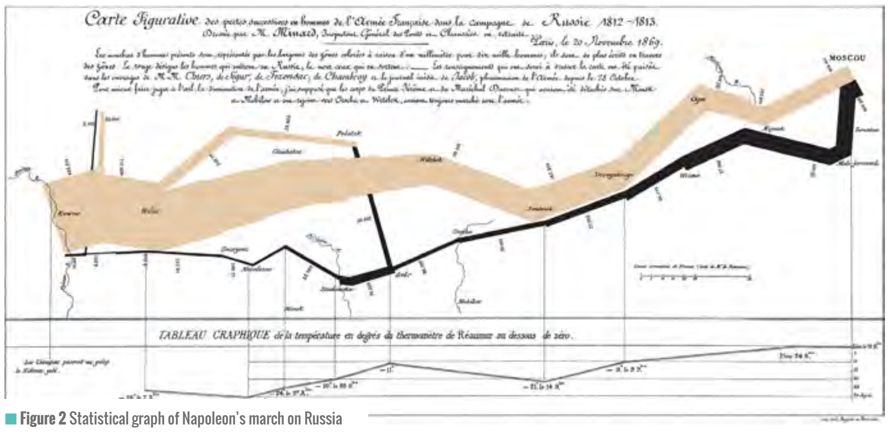


Figure 2 Statistical graph of Napoleon's march on Russia

Effecting change with data

As well as telling a story with data, visualisations can be starting points to bring about change or to help gain support for a cause. The next historical example dates back to 1854, when there was an outbreak of cholera in the Soho area of London. At the time, it was a commonly held belief that cholera was caused by bad air in the area. A physician named John Snow held an alternative belief that cholera was being transmitted by a contaminated water supply. To help prove this theory, he mapped the deaths from cholera in the Soho area. The map revealed that the deaths were centred around Broad Street, and the residents there were getting their water from the pump on this street.

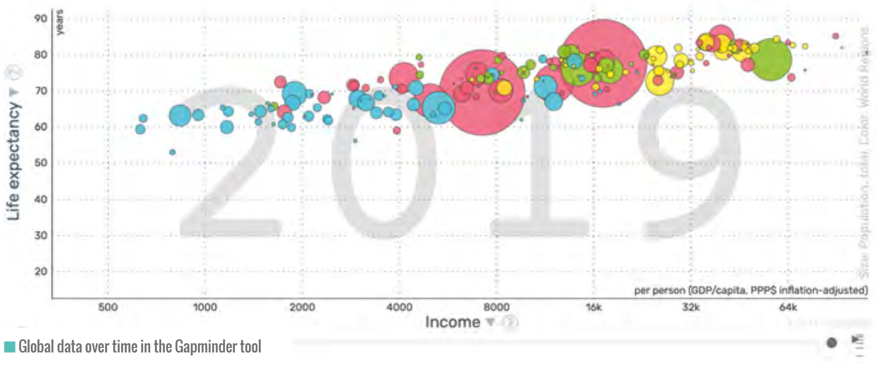
By visualising the data in this way (now known as a dot map, a map type that uses a dot symbol to show the presence of a feature or phenomenon), Snow was able to convince the local council to disable the water pump. It is widely recognised that this visualisation helped to save many lives.

Getting hands-on with interactive visualisations

Moving from the past to the present day, the final point I would like to highlight is a free visualisation tool by Gapminder (**helloworld.cc/gapminder**). This software allows you to view global data and see how it has changed over time. The default data in the bubble graph compares life expectancy with income. By using the slider and clicking on the 'Play' button, you can see how this has changed from the year 1799 to the present day.

The data on the x- and y-axes is customisable, allowing you to compare a huge range of other factors, such as CO₂ emissions and educational standards. I like this tool, compared to the historical examples I've discussed, as it is much more hands-on for students, and presents opportunities to discuss concepts such as trends, correlation, and outliers.

After allowing students to see the data available to them, a good activity would be to get the class to think about what factors make a country a great place to live. They can then use the Gapminder tool to investigate which country best fits their ideals.


Life expectancy
years
90
80
70
60
50
40
30
20
2019
500
1000
2000
4000
8000
16k
32k
64k
Income
per person (GDP/capita, PPP$ inflation-adjusted)
Size: Population, total Color: World Regions

Global data over time in the Gapminder tool

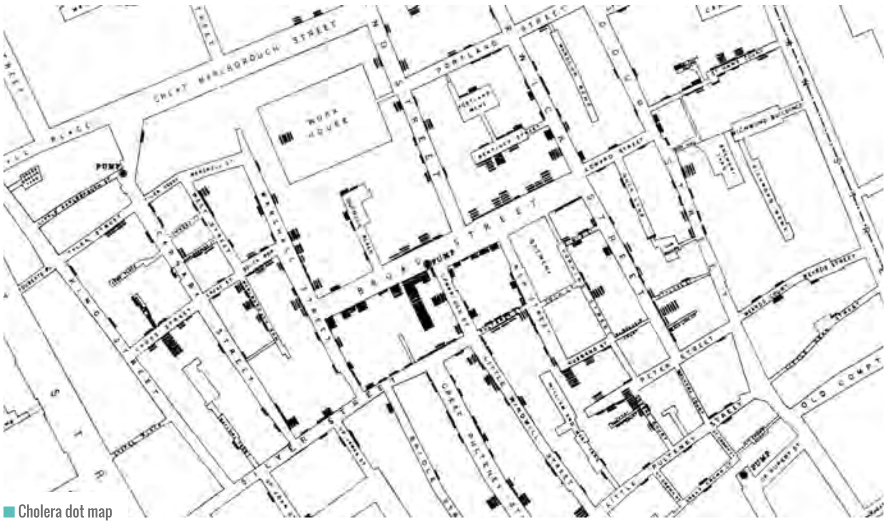
Practical suggestions for the classroom

As suggested, the Gapminder tool is a good place to start to get students exploring data sets. You can also use the Turner's Graph of the Week website (**helloworld.cc/turner**) to reinforce the idea of using visualisations to spot trends and correlations, as well as to make predictions and gain insights. The site provides free worksheets that include visualised data and sets of questions to prompt analysis of the data, perfect for in-class activities and homework.

I'd recommend allowing time for your students to explore the visualised data sets on the Information is Beautiful website, including data talking points such as 'riskiest activities during the pandemic', and how much music streaming services pay their artists (**helloworld.cc/information**). Another favourite of mine is Dear Data (**helloworld.cc/deardata**), a project between two information designers to collect data about their lives each week and visualise it on a postcard sent to each other. A nice follow-up activity to the Dear Data website would be to ask students to spend a week collecting and visualising data about their own lives. For example, they could monitor their mobile phone or app usage, or what food they've eaten. Once they have completed the activity, ask them to write a short report on what they have learnt from their visualised data and how they might use the information to make changes in their lives.

Finally, data science is not just the remit of computing. Do explore the cross-curricular opportunities that you could take advantage of when teaching visualisations; the examples in this article have clear links with history, geography, and maths.

Using some of these examples and tools, I hope your students will have understood that visualisations can be much more creative than the traditional bar and pie charts we're all used to seeing, and that data has the power to tell a story, share information, and bring about change. (HW)



Cholera dot map

BEN GARSIDE

Ben works for the Raspberry Pi Foundation and is also a Computing at School community leader. Recently, Ben has been developing resources for the Teach Computing Curriculum, as well as writing online courses and content for Isaac Computer Science (**@BenberryPi**).

USING DATA TO OPTIMISE A SCHOOL GARDEN

Chris Aviles describes how his class uses data to make their gardens more productive

As a society, we have never before collected more data about individuals. Despite this, most people and institutions do a poor job of interpreting data and using it to make meaningful change. I wanted to tackle this problem on a local scale with my learners in FH Grows.

FH Grows is the name of my seventh-grade class (aged 12–13), and a student-run agriculture business at Knollwood Middle School in Fair Haven, New Jersey. In FH Grows, we grow produce and sell it, both online and through student-run farmers' markets. Any produce we don't sell is donated to our local soup kitchen. To get the most out of our school gardens, students have built sensors and monitors using Raspberry Pis. These sensors collect data, which allows me to teach students how to get better at interpreting data themselves, and how to turn it into action.

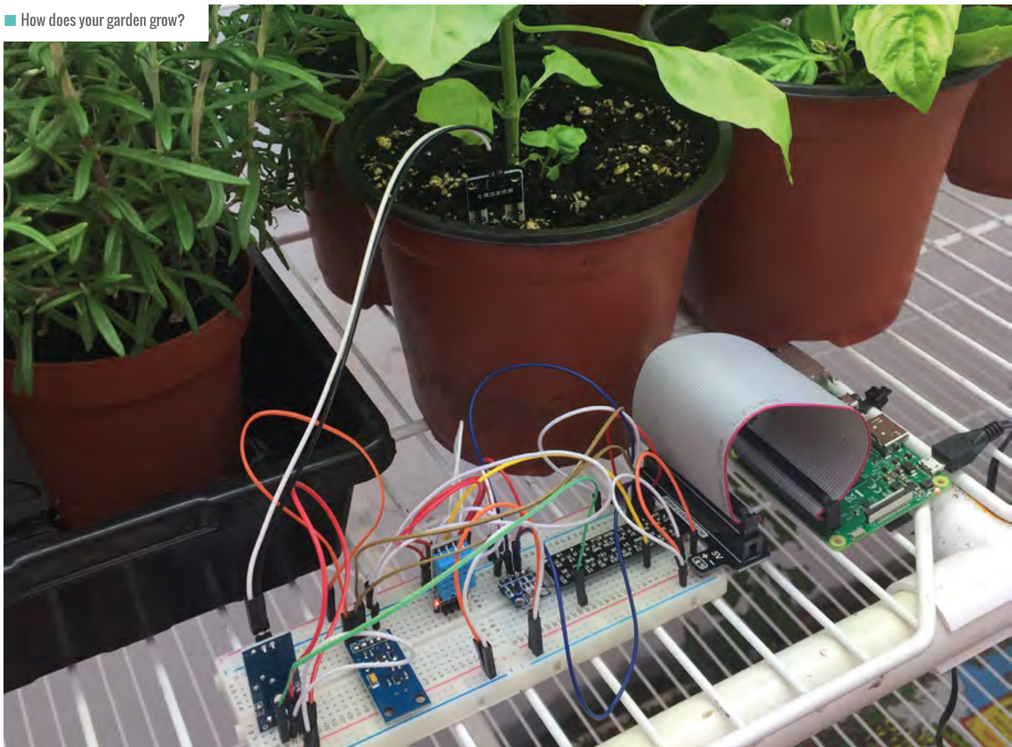
Turning data into action

In the greenhouse, our gardens, and our alternative growing stations (we have hydroponics, aquaponics, and aeroponics) we use sensors to log temperature, humidity, and other important data points. This data is streamed in real time on a site I created for the class. When students come into the classroom, one of the first things we do is look at the live data on the site and find out what is going on in our gardens. Over the course of the semester, students are taught about ideal growing conditions. If, when we look at the data, we see that conditions aren't ideal, it's time to get to work.

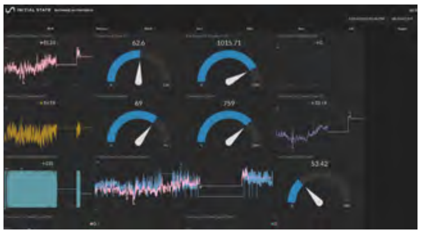
If we see that the greenhouse is too hot, over 85 degrees Fahrenheit, students go and open the greenhouse door. We check the temperature a bit later, and if it's still too hot, students turn on a fan. But how many fans do we need to turn on? After experimenting, we know that each fan lowers the greenhouse temperature by between 7 and 10 degrees. Opening the door and turning on both fans can bring a greenhouse that's pushing 100 degrees in late May or early June down to a more manageable 80 degrees.

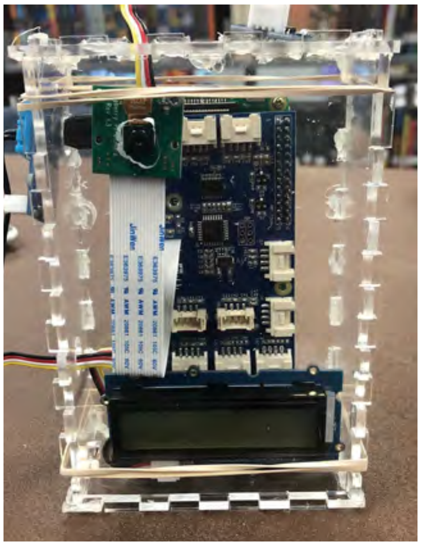
Turning data into action can allow for some creativity as well. Overwatering plants can be a real problem; we found that our plants were turning yellow because we were watering them every day when we didn't need to. How could we solve this problem and become more efficient at watering? Students built a Raspberry Pi that used a moisture sensor to find out when a plant needed to be watered. We used a plant with this moisture sensor in the soil as our control plant. We figured that if we watered the control plant at the same time as we watered all our other plants, when the control plant was dry (giving a negative moisture signal), the rest of the plants in the greenhouse would need to be watered as well.

This method of determining when to water our plants worked well. We rarely ever saw our plants turn yellow from overwatering again. Here is where the creativity came in. We received a signal from Raspberry Pi when the soil was not wet enough, so we played around with what we could do with that signal. We displayed it on the dashboard along with our other data, but we also decided to make the signal send us an email from the plant. When I showed students how this worked, they decided to write the message from the plant in the first person. So every week or so, we received an email from Carl the Control Plant asking us to come out and water him!


How does your garden grow?

If students didn't honour Carl's request for water, if they didn't use data to know when to cool the greenhouse, or if they had not done the fan experiments to see how much cooler their measures would make the greenhouse, all our plants, such as the basil we sell to the pizza places in town, would die. This is the beauty of combining data literacy with a school garden: failure to interpret data, then take action based on their interpretation, has real consequences: our produce could die. When it takes 60–120 days to grow the average vegetable, the loss of plants is significant. We lose all the time and energy that went into growing those plants, as well as all the revenue they would have brought us. Furthermore, I love the urgency that combining data and the school garden creates, because many students have learnt the valuable life lesson that not making a decision is also making a decision. If students freeze or do nothing when confronted with the data about the garden, that too has consequences.





Our Pi prediction system has given us far more accurate data

> FAILURE TO INTERPRET DATA THEN ACT ON THEIR INTERPRETATION HAS REAL CONSEQUENCES: OUR PRODUCE COULD DIE

Using data to spot trends and make predictions

The other major way we use data in FH Grows is to spot trends and make predictions. This is different to using data to create the ideal growing conditions in our garden every day; the sensors we use also provide a way for us to use information about the past to predict the future. FH Grows has about two years' worth of weather data from our Raspberry Pi weather station (there are guides available online if you wish to build a weather station of your own). By collecting weather data year after year, we can start to determine important information such as the best time to plant our veggies.

For example, one of the most useful data points on our Raspberry Pi weather station is the ground temperature sensor. Last semester, we wanted to squeeze in a cool-weather grow in our garden. This post-winter grow can be done between March and June if you time it right. Getting an extra growing cycle from our garden is incredibly valuable, not only to FH Grows as a business (as we would be growing more produce to sell), but as a way to get an additional learning cycle out of the garden.

So, using two seasons' worth of ground temperature data, we set out to predict when the ground in our garden would be cool enough to do this cool-veggie grow. Students looked at the data we had from the weather station and compared it to different websites that predicted the last frost of the season in our area. We found that the ground right outside our door warmed up two weeks earlier than the more general prediction given on weather websites. With this information, we were able to get a full cool-weather grow at a time where our garden used to lie dormant.

We also used Raspberry Pi to help us predict whether or not it was going to rain over the weekend, by using it to connect to Weather Underground and looking at data from previous years. If we believed it would not rain over the weekend, we would water our gardens on a Friday. If we thought it would rain, we let Mother Nature water our gardens for us. Our prediction, using the Pi and previous data, was more accurate for our immediate area than the general weather reports you would get on the radio or an app, as those considered a much larger area when making their predictions.

It seems that we are going to be collecting even more data in the future, and it is important that we get our students comfortable working with it. The school garden supported by Raspberry Pi's amazing ability to collect data is a boon for any teacher who wants to help students learn how to interpret data and turn it into action. (HW)

CHRIS AVILES
Chris is a Raspberry Pi Certified Educator and a teacher at Knollwood Middle School in the Fair Haven school district in New Jersey, USA. There, he runs the renowned Fair Haven Innovates program he created in 2015.

TEACHING DATABASES USING BIG DATA

Databases are fundamental to modern society, so why do many young people (and their teachers) find the topic less than engaging?

There has been major curricular change in Scottish computing education over the last eight years. Previously, there were separate qualifications in computing and information systems. To generalise, the former focused on programming and the second on databases. The new qualifications, introduced in 2014, combined these two separate qualifications into one new qualification called 'computing science'. A more recent update introduced SQL queries at all levels of computing science.

Soon after this update, at a meeting with local computing teachers, we were discussing, first, the importance of databases in the study of computing science and, second, how it can be difficult to engage pupils in a topic that can be drier than programming or designing websites.

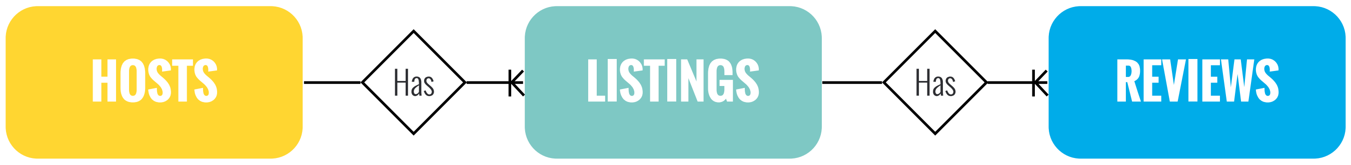
How can a topic that underpins so much of modern technology be seen as boring or irrelevant by pupils who, unwittingly or not, interact with huge data sets multiple times in a day? In this article, I'd like to explore an approach to learning about databases that show pupils the relevance of understanding and being able to extract meaning from modern data sets.

Big data

Imagine an introductory database lesson that talks about the millions and billions of records and fields contained in the Amazon or Twitter databases. Pupils, keen to get insight into the technology behind these huge enterprises, are then directed to Microsoft Access, where they have to create a single table, declare a few fields, and create a handful of records in an address book or music database. This discrepancy, between the reality of huge, enterprise-scale databases and what pupils can actually create, is massive. How many of us have asked pupils to enter five or ten records into a database, with no actual thought as to the educational value? Why do we need a database to store ten records? A spreadsheet, or even a table in Word, would allow us to store and see this information at a glance.

To see the need for databases, and their enormous power and value, we must use big data: data sets with thousands of records, data that requires queries and sorts in order to tell us something meaningful. Multiple, linked tables can then be used to show why careful design of databases is so important. Close to home for me, for example, Glasgow City Council and the Scottish Government publish a range of data sets about crime, education, environment, and population, which can be used in class (**helloworld.cc/glasgowdata** and **helloworld.cc/scotlanddata**).

The Scottish Index of Multiple Deprivation is an excellent, interactive example of data being used to analyse and compare huge amounts of socioeconomic data (**helloworld.cc/smd**). Simply comparing Glasgow and Edinburgh shows stark differences, but



Inside Airbnb offers vast amounts of data relating to hosts, listings, and reviews

Field(s)	listing_url, summary, price, bedrooms
Table(s)	Listings
Search criteria	Bedrooms > 1, price <=60, zipcode=EH99%
Sort order	review_scores_rating Desc

Figure 1 By identifying which tables are being queried, you can spot any joins that are required. Thinking at this stage about fields, criteria, and ordering leads naturally to writing the query in SQL

what can we say about the educational, health, and employment opportunities for children growing up in rural versus urban areas? Drumchapel and Bearsden are neighbouring areas of Glasgow, but have vastly different opportunities for young people growing up just a few streets apart.

Airbnb

Another powerful example is the big data available about Airbnb. Airbnb has disrupted the world of travel and accommodation, and the site Inside Airbnb (**helloworld.cc/airbnb**) allows you to download vast amounts of data that has been scraped from the site for many of the most popular destinations in the world. This context allows us to see the real-world website and the database behind it.

These data sets require some analysis and work before they can be used in a classroom situation. Using the Edinburgh Airbnb data (**helloworld.cc/edinburghairbnb**), I downloaded listings, hosts, and review data for thousands of destinations in the capital and imported it into Access. Some fields needed a little work to make them usable. Creating a relational database of the data allows us to illustrate the cardinality of the data.

This data is too much to take in just by looking through the tables, but we can now start to construct queries. If I wanted to stay near the Scottish Government building in the EH99 postcode area, but could only afford £60 a night for at least two bedrooms, which table or tables would I look in, and what criteria would I use? The SQA exam board that my school follows plans queries as shown in **Figure 1**. From there, it's then trivial to code this query:

```
SELECT listing_url, summary,
price, bedrooms
FROM Listings
WHERE bedrooms>1 AND price 0
<=60 and zipcode LIKE 'EH99%'
ORDER BY review_scores_rating
DESC;
```

Looking at a single listing on the Airbnb website will show you the listing, some host information, and a summary of their reviews. What better illustration of a join? This view is created dynamically using data from all three tables.

Data on Inside Airbnb is updated regularly. Pupils can click on the URLs contained in the database and verify that the data they're examining is for a real host with a real listing in a real city. I believe this authenticity, and the quantity of data used, will start to show pupils the value of database management software. The focus on SQL will mean they don't become experts in Access but, instead, come to understand the underlying structure of databases and the language used to query the data.

Other discussions will follow. Is it legal to scrape data? None of the downloaded data is hidden or hacked, but do we have the right to extract, store, and manipulate it? How could bot software be programmed to scrape the data? Why might the web server view the scraper software as an attack?

Then we can look at the social and economic implications of Airbnb. Would you like to live alongside someone who lets out their apartment to multiple people 250 nights of the year? What impact has the site had on the hotel industry? What impact does it have on cities that welcome lots of tourists each year? Does Airbnb drive up rents and restrict availability for local residents?

There are lots of other sources of large data sets out there. I hope this article provides you with a helpful starting point for finding examples that will grab the attention of the young people in your classes, which they'll want to manipulate and manage. (HW)

“ TO SEE THE POWER AND VALUE OF DATABASES, WE MUST LOOK TO DATA SETS WITH THOUSANDS OF RECORDS

IMDB

Many of our young people will be big fans of online streaming services, and will likely be familiar with the Internet Movie Database (IMDb). Data sets from IMDb and its vast library of films and television series can be found at **datasets.imdbws.com**, with information about the data at **imdb.com/interfaces**. This could be an engaging context to explore and manipulate.

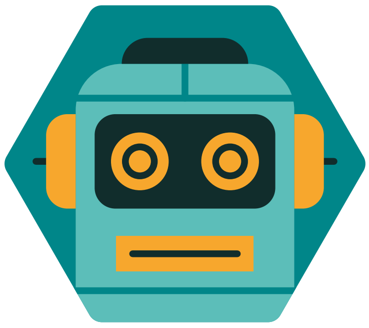
TONY HARKINS
Tony is head of computing at St Aloysius' College, Glasgow.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) encompasses a range of technologies that analyse large data sets to identify patterns, which they then use to make predictions, classify objects, or generate entirely new artefacts.

In developing an understanding of AI, learners will explore examples of different AI applications and the crucial role of data in these systems. In comparison to traditional rule-based systems, where decisions made by the system are governed by programmed rules, AI systems make decisions based on patterns in data — patterns that may be harder for humans to explain. Learners will explore how AI systems make use of models, as well as looking at how such models are generated. Alongside this technical knowledge, learners will also need to understand the social and ethical implications of AI.

Artificial intelligence is an emerging technology, and as such is a new strand within the Raspberry Pi Foundation's taxonomy of formal education content. While there is a broad consensus as to the importance of learners understanding, applying, and building with AI systems, there is less agreement about what, how, and when we should teach it. Many relevant tools, initiatives, and resources are nonetheless available, and there are multiple active research projects in this area. This section therefore provides an insight into AI education at this point in time.



IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** artificial intelligence, in summary
- **What the research says:** AI ethics
- **What the research says:** big ideas in AI education
- AI for preschoolers
- The history of AI
- Ideas for discussions in the classroom

IN SUMMARY

ARTIFICIAL INTELLIGENCE

Understand the capabilities and limitations of artificial intelligence, along with its applications and wider impacts

STAGE 1

- Recognise that AI systems are computing systems that are designed and built by people
- **Describe examples of the problems associated with using and trusting AI systems**
- Describe the purpose of familiar AI applications

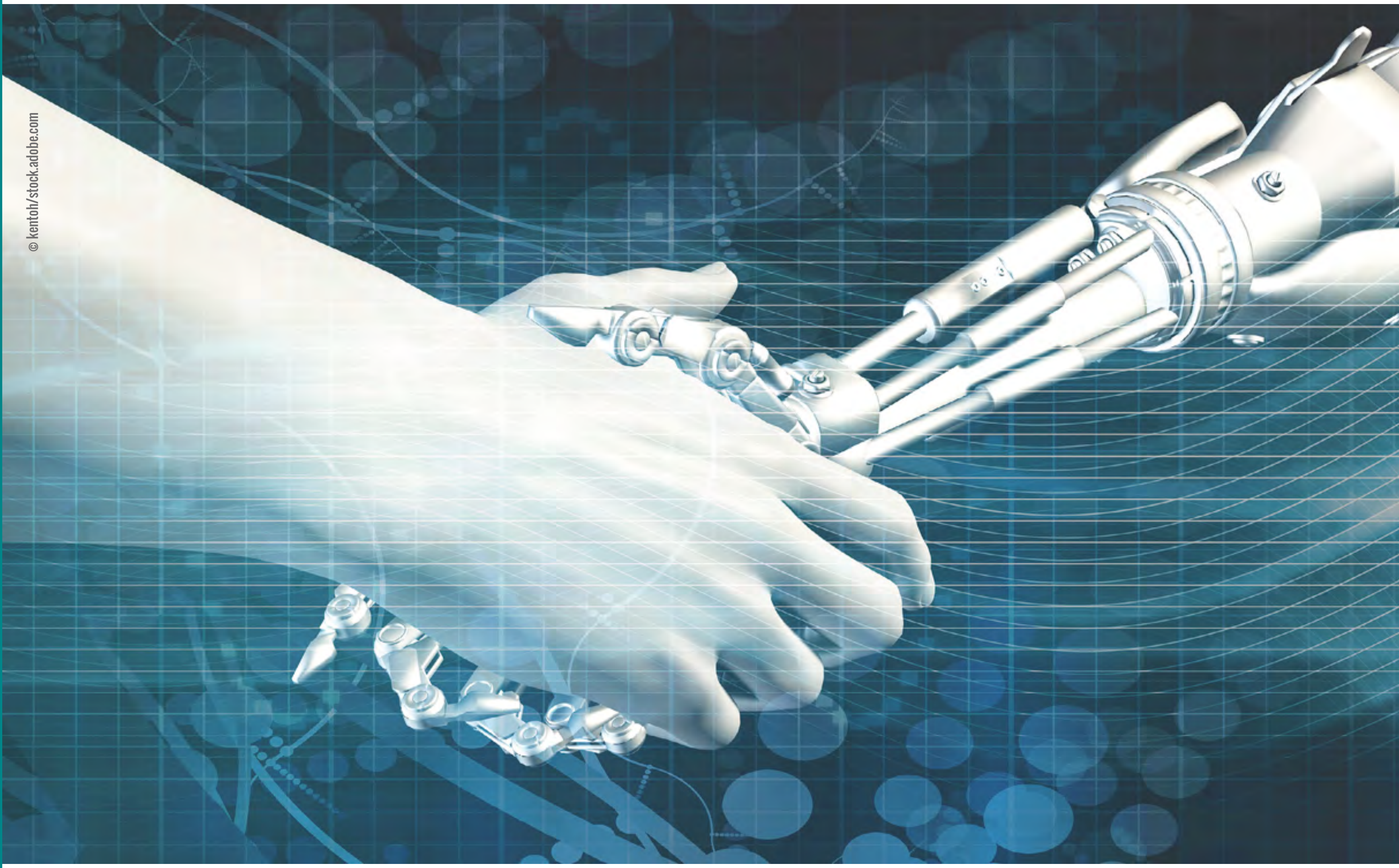
STAGE 2

- Name examples of AI applications in a wider societal context
- **Contrast the benefits and issues of using and trusting AI systems**
- Describe how data is key to AI systems and the decisions they make
- **Describe examples of where AI can generate digital artefacts**
- Describe examples of where AI can classify (images, sounds, text, etc.)
- **Understand that AI systems can be used to make predictions**
- Describe a model as a representation of the real or a fictional world
- **Recognise that a model's performance can be improved by adding more training data**

In the table below, you will find suggested learning outcomes associated with the 'Artificial intelligence' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally proposed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Describe how the quality of data determines the success of an AI application	Describe how a machine learning model is trained	Describe the main AI paradigms
Explain how AI systems pose a potential threat to equal opportunities	**Name ethical standards and guidelines for creating and using AI**	**Describe the potential social, cultural, and economic impacts of AI**
Identify common types of AI application	Compare the advantages and disadvantages of supervised learning algorithms	Compare AI learning types (supervised, unsupervised, reinforcement)
Identify the parts of a system that are AI and the parts that are not	**Design and test supervised learning solutions for classification problems**	**Compare AI task types (classification, regression, clustering, generative, decision-making)**
Compare data-driven models and rule-based models	Train models and incorporate them into a programmed solution	Identify different AI engines (e.g. decision trees, *k*-nearest neighbors, neural networks, linear regression)
Describe how the data life cycle is applied to an AI system	**Evaluate whether a model is fit or not fit for purpose**	**Explain that different engines have different levels of explainability**
Train a machine learning model	Identify a neural network as a supervised learning algorithm	Choose the right algorithm to solve a particular problem
Explain the difference between training and test data	**Identify the different components of a neural network and describe their purpose**	**Describe the role of weights and backpropagation during the training of a neural network**
Evaluate the performance of a decision-tree model	Evaluate the performance of a neural network	



ENGAGING CHILDREN WITH AI ETHICS

STORY BY Ben Garside

Artificial intelligence (AI) is a term that is now in most people's vocabulary, even if the meaning isn't always understood. With huge amounts of data constantly being generated in the world, alongside regular developments in hardware capabilities, more and more systems that we interact with, or that make decisions about us, use AI. As a result, educators are becoming increasingly interested in teaching young people about this technology. At the same time, there isn't a huge amount of evidence-based research in this area of education, so curriculum designers are making choices as how best to approach teaching young people about it.

Researchers Mhairi Aitken and Morgan Briggs at The Alan Turing Institute have found that many curricula designs focus on AI and data science skills and capabilities, to "equip the next generation to pursue careers in these fields" (**helloworld.cc/aitken2022**). They make the case that AI education needs to go beyond simply preparing young people for careers, and that we should instead make it a priority to encourage discussions about the social and ethical ways in which AI is designed, developed, and deployed.

Why is this important?

Aitken and Briggs stress that learning about AI through this lens is important; children need to understand the role of AI in their lives and critically engage with this technology if they are to make informed choices about how they interact with it. The paper illustrates that it's almost guaranteed that every young person will at some point interact with, or be the subject of, a decision made by an AI system. For example, if a child interacts with a voice-based digital home assistant, or uses a social media app, they will be submitting their data and interacting with AI applications, perhaps unbeknown to them! These examples might both seem harmless, but many such systems aren't designed with children in mind, and without children having a basic education about how they work, there may be unforeseen problems. Aitken and Briggs quote the case of a ten-year-old child who asked an Amazon Alexa for a challenge, with Alexa responding with a challenge that placed the child's safety at risk.

The paper also describes how AI systems might have an impact on "shaping children's views of the world", such as deciding what content to display to them online, what content to filter out, and influencing the friendships they develop via social media. AI could also "impact and shape children's lives" through influencing the provision of services. A recent example Aitken and Briggs point to was the algorithm used by Ofqual (England's Office of Qualifications and Examinations Regulation) to help determine grades after examinations were cancelled in 2020 due to the pandemic. The algorithm awarded exam grades which clearly showed that students from less privileged backgrounds were disadvantaged compared with students who attended private schools. The system was designed to achieve fairness, but because of a lack of attention towards ethical considerations, the result was a system that "exacerbated existing inequalities in society leading to unfair outcomes". AI is clearly increasingly impacting children's lives and shaping the future societies in which they will live and work. Aitken and Briggs conclude that it is therefore vital that children and young people are equipped to interrogate and understand the role of AI systems.

"AI IS INCREASINGLY IMPACTING CHILDREN'S LIVES AND SHAPING THE FUTURE SOCIETIES IN WHICH THEY WILL LIVE AND WORK

Child-centred AI

As well as the more negative examples discussed above, the paper highlights that "children and young people have a unique set of needs, and it is important to note that if developed ethically and responsibly and with children's voices included and listened to, AI technologies could provide beneficial outcomes". For example, the researchers argue that within education, AI could help support children's learning, such as through real-time translation, allowing children to access global educational resources, or by supporting those with visual or hearing impairments. However, to ensure that children's needs and interests are accounted for, ethical principles need to be followed. Here, Aitken and Briggs introduce an area of research called child-centred AI, which aims to ensure that children are involved throughout all stages of the AI life cycle in a "meaningful and worthwhile way". A summary of the main components of child-centred AI are as follows:

- Helping children to make informed choices about their interactions with, and uses of, AI
- Enabling children and young people to play a role in discussions shaping future AI practices
- Ensuring the next generation of developers and policymakers are equipped with an understanding of the ethical considerations around AI and its uses
- Ensuring ethical mindsets of future developers and members of the tech industry

Organisations that are creating and using AI may find it challenging to put child-centred AI into practice. The paper thus recommends a substantive approach to engaging children around AI ethics, engaging young people in discussions that help them understand the role and impact AI has in their lives and allows them to "critique the ways that AI is designed, developed, and deployed". The researchers focus on the benefits of this type of approach and highlight that if young people are involved in this way, they are better equipped to make "informed choices" and to "hold AI systems and their developers to account". Translating this for the classroom, this means that to engage children with AI, we should begin with no assumptions about what they already know, and should promote a dialogue, asking questions like:

- What do you know about AI? What do you want to know?
- What are your concerns, interests, and priorities?
- What are the important issues in your lives to which AI might relate, or have a positive or negative impact?

As the paper concludes, it is essential for future AI workforces to include a "diverse mix of skills and expertise encompassing technical, social, ethical, legal, and policy dimensions". While there is a place for teaching young people the technical skills related to AI, children and wider society would clearly benefit from education going beyond this and teaching them about the importance of ethics and critical thinking within this ever-changing technology. (HW)

FURTHER READING

✓ **Aitken, M., & Briggs, M. (2022). Engaging children with AI ethics. In *AI, data science, and young people. Understanding computing education (Vol 3). Proceedings of the Raspberry Pi Foundation Research Seminars.*** **helloworld.cc/aitken2022**

BIG IDEAS IN AI EDUCATION

STORY BY Sue Sentance

From September 2021 to March 2022, the Raspberry Pi Foundation hosted a series of seminars in partnership with The Alan Turing Institute focused on artificial intelligence (AI), machine learning, and data science education (helloworld.cc/AIseminars). These are important topics in both the Foundation's learning resources for learners and educators, and for our programmes of research, and will only become more important as AI increasingly becomes ingrained in our societies. In this article, I will summarise and explore some of the ideas shared in one of these seminars, presented by Professor Dave Touretzky and Professor Fred Martin, about how to approach teaching AI.

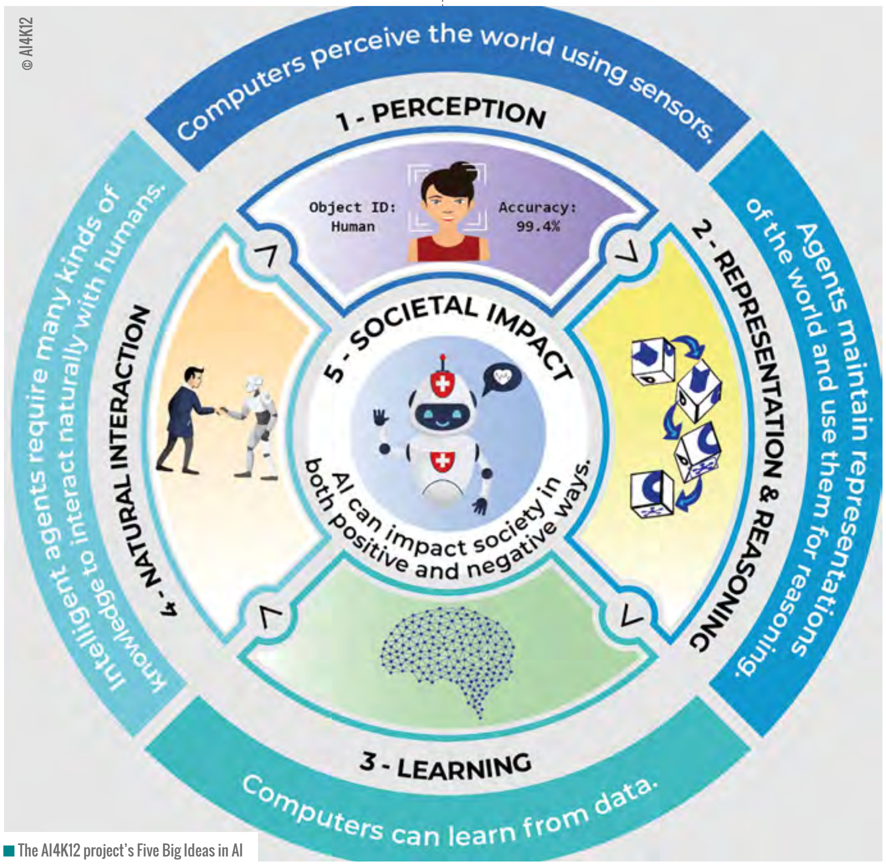
AI4K12

The AI4K12 project (**ai4k12.org**), spearheaded by Touretzky and Martin, focuses on teaching AI in K–12 (that is, to learners aged 4–18) in the US. The AI4K12 team has aligned its vision for AI education to the CSTA standards for computer science education (**helloworld.cc/CSTAstandards**). These standards, published in 2017, describe what educators should teach in US schools across the discipline of computer science, but they say very little about AI. As such, this was the stimulus for starting the AI4K12 initiative.

The AI4K12 project has a number of goals. One is to develop a curated resource directory for K–12 teachers, and another is to create a community of K–12 resource developers. Several members of the AI4K12 working group are practitioners in the classroom who have made a huge contribution to taking this project from idea stage to fruition. If you've heard of AI4K12 before, it's probably because of the Five Big Ideas the team has set out, to encompass the AI field from the perspective of school-aged children (**helloworld.cc/fivebigideas**). These ideas are:

1. **Perception:** the idea that computers perceive the world through sensing
2. **Representation and reasoning:** the idea that agents maintain representations of the world and use them for reasoning
3. **Learning:** the idea that computers can learn from data
4. **Natural interaction:** the idea that intelligent agents require many types of knowledge to interact naturally with humans
5. **Societal impact:** the idea that artificial intelligence can impact society in both positive and negative ways

We sometimes hear concerns that resources being developed to teach AI concepts to young people are too narrowly focused on machine learning, particularly supervised learning for classification. It's clear from the AI4K12 Five Big Ideas that the team's definition of the AI field encompasses much more than this one


© AI4K12
Computers perceive the world using sensors.
1 - PERCEPTION
Object ID: Human
Accuracy: 99.4%
2 - REPRESENTATION & REASONING
Agents maintain representations of the world and use them for reasoning.
3 - LEARNING
Computers can learn from data.
4 - NATURAL INTERACTION
Intelligent agents require many kinds of knowledge to interact naturally with humans.
5 - SOCIETAL IMPACT
AI can impact society in both positive and negative ways.
The AI4K12 project's Five Big Ideas in AI

area. Despite being developed for a US audience, I believe the description laid out in these five ideas is immensely useful to all educators, researchers, and policymakers around the world who are interested in AI education.

During the seminar, Touretzky and Martin shared some great practical examples. Martin explained how the big ideas translate into learning outcomes for each of the four age groups (ages 5–8, 9–11, 12–14, and 15–18). You can find out more about their examples in their presentation slides (**helloworld.cc/AI4K12ppt**) or the seminar recording (**helloworld.cc/AI4K12seminar**).

I was struck by how much the AI4K12 team has thought about progression — what you learn when, and in which sequence — which we do really need to understand well before we can start to teach AI in any formal way. For example, looking at how we might teach visual perception to young people, children might start when very young by using a tool such as Teachable Machine to understand that they can teach a computer to recognise what they want it to see (**helloworld.cc/teachablemachine**), then move on to building an application using Scratch plug-ins or CalypsoAI (**calypsoai.com**), and then to learning the different levels of visual structure and understanding the abstraction pipeline — the hierarchy of increasingly abstract things.

WE MUST UNDERSTAND PROGRESSION — WHAT YOU LEARN WHEN, AND IN WHICH SEQUENCE — BEFORE WE CAN TEACH AI

Glass and opaque boxes

Touretzky and Martin support teaching AI to children using a glass-box approach. By this we mean that we should give students information about how AI systems work, and show the inner workings, so to speak. The opposite would be an opaque-box approach, which would mean showing students an AI system's inputs and outputs only, to demonstrate what AI is capable of without trying to teach any technical detail.

The AI4K12 researchers are keen for learners to understand, at an age-appropriate level, what is going on inside an AI system, not just what the system can do. They believe it's important for young people to build mental models of how AI systems work, and that when young people get older, they should be able to use their increasing knowledge and skills to develop their own AI applications.

What does AI thinking look like?

Touretzky addressed the question of what AI thinking looks like in school. His approach was to start with computational thinking (he used the example of the Barefoot project's description of computational thinking as a starting point; **helloworld.cc/barefootCT**) and describes AI thinking as an extension that includes the following skills:

- Perception
- Reasoning
- Representation
- Machine learning
- Language understanding
- Autonomous robots

He went described AI thinking as furthering the ideas of abstraction and algorithmic thinking commonly associated with computational thinking, stating that with AI, computation actually is thinking. My view is that to fully define AI thinking, we need to dig a bit deeper into, for example, what is involved in developing an understanding of perception and representation.

Thinking back to a previous Raspberry Pi Foundation research seminar, Professor Matti Tedre and Dr Henriikka Vartiainen shared their description of computational thinking 2.0 (**helloworld.cc/tedreseminar**). Their description focuses only on the 'Learning' aspect of the AI4K12 Five Big Ideas, and on the distinct ways that thinking underlies data-driven programming and traditional programming. From this, we can see some differences between how different groups of researchers describe the thinking skills young people need in order to understand and develop AI systems. Tedre and Vartiainen are working on a more granular description of machine learning thinking, which has the potential to impact the way we teach machine learning in school.

Another description of AI thinking comes from Juan David Rodríguez García, who presented his system, LearningML, at another of the Raspberry Pi Foundation's seminars (**helloworld.cc/garciaseminar**). Rodríguez García drew on a paper by Brummelen, Shen, and Patton (**helloworld.cc/brummelen2019**), who extended Brennan and Resnick's CT framework of concepts, practices, and perspectives (**helloworld.cc/brennan2012**) to include concepts such as classification, prediction, and generation, together with practices such as training, validating, and testing.

What I take from this is that there is much still to research and discuss in this area! It's a real privilege to be able to hear from experts in the field and compare and contrast different standpoints and views.

*Read more from our AI and data science education presenters in their write-ups of their seminars (**helloworld.cc/RPFseminarpapers**).* (HW)

FURTHER READING

✓ Touretzky, D. S., & Martin, F. (2022, January 11). *Teaching Artificial Intelligence in K-12* [seminar presentation]. Raspberry Pi Research Seminar. **helloworld.cc/touretzky2022**

POPBOTS OPEN AI TO THE YOUNGEST LEARNERS

From Siri to gaming, artificial intelligence is all around us. **Ben Hall** explores an innovative project that makes this technology accessible to preschoolers

Very often, the most vivid examples of how quickly children learn and progress can be seen in the youngest age groups. A great example is how children use construction-based resources such as wooden blocks, train tracks, or LEGO®. Often, children experience these for the first time when they enter the primary classroom. Initially, they will use these resources in quite a free, unstructured way. Over time, with guidance and support, they construct increasingly complex and imaginative models, often linked to other areas of learning. The models from the end of the year are unrecognisable from those constructed at the beginning.

Can the same be said of children's first encounters with technology in the classroom? Many schools will provide access to tablets or computers as part of their early-years provision, but how much of that is about consuming the technology, and do children relate it to real-world applications? In my experience, simply having the devices available is not enough — most children start school able to navigate their way around a tablet, and merely exposing them to it will not introduce new concepts. How can we develop children's computing knowledge so it will encourage them to be more than just passive users?

WHAT IS A POPBOT?

PopBots are constructed using LEGO and use a mobile phone with additional LEGO or Arduino peripherals. Users interact with them via a programming interface on a tablet or computer.

AI literacy

Could artificial intelligence (AI) provide an answer? A Massachusetts Institute of Technology team has aimed to make AI accessible by developing PopBots, small robots that are used to introduce AI to children. They have also developed easy-to-use resources to complement the hardware.

One such resource explores machine learning, where a robot can be programmed to sort healthy or unhealthy foods. Children begin by classifying foods for the robot, but soon realise it would take too long to do this for every single food type. Through supervised machine learning, children can quickly train PopBots to classify foods and develop their own understanding of healthy eating. Through this simple activity, children become AI-literate creators — turning a passive device into something that makes informed, intelligent decisions.

Randi Williams, who worked on the PopBots project, gives her view on introducing children to AI: "Children's views of themselves relative to technology change. Their views of how much they can participate in technological invention change. I love the fact that early AI education makes children feel more curious about their world, and empowered to change it."

Research-led

PopBots are still in the early stages of development and are not yet widely available. Despite this, there is already a growing body of research investigating the inclusion of AI in the curriculum at an early age. Williams would like to see PopBots developed as an open-source platform that students could build from classroom materials. Research supports further development.

A recent paper by Williams and colleagues investigates how AI can influence young children's perceptions of robots (**helloworld.cc/williams2019**). They found perceptions of robots are shaped at an early age, so for children to be AI-literate, their earliest experiences should be meaningful and informed. It is an emerging technology, so the research is at an early stage, but there is no doubt that AI will increasingly shape our world. Helping children develop a conceptual understanding at an early age needs to be at the forefront of curriculum development. (HW)

BEN HALL
Ben is a learning manager at the Raspberry Pi Foundation. He's a former primary teacher, a CAS Master Teacher, and a Raspberry Pi Certified Educator (**@hengehall**).

SNAPSHOTS FROM THE HISTORY OF AI

George Boukeas introduces fascinating stories to share with your students

The story of artificial intelligence (AI) is a story about humans trying to understand what makes them human. Some of the episodes in this story are fascinating, and could help learners catch a glimpse of what the field is about and, with luck, compel them to investigate further.

The imitation game

In 1950, Alan Turing published a philosophical essay titled *Computing Machinery and Intelligence*, which started with the words: "I propose to consider the question: can machines think?" Yet Turing did not attempt to define what it means to think. Instead, he suggested a game as a proxy for answering the question: the imitation game.

In modern terms, you can imagine a human interrogator chatting online with another human and a machine. If the interrogator does not successfully determine which of the other two is the human and which is the machine, then the question has been answered: this is a machine that can think.

This imitation game is now a fiercely debated benchmark of artificial intelligence called the Turing test. Notice the shift in focus that Turing suggests: thinking is to be identified in terms of external behaviour, not in terms of any internal processes. Humans are still the yardstick for intelligence, but there is no requirement that a machine should think in the same way humans do, as long as it behaves in a way that suggests some sort of thinking to humans.

In his essay, Turing also discusses learning machines. Instead of building highly complex programs that would prescribe every aspect of a machine's behaviour, we could build simpler programs that would prescribe mechanisms for learning, and then train the machine to learn the desired behaviour. Turing provides an excellent metaphor that could be used in class to describe the essence of machine learning: "Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. We have thus divided our problem into two parts. The child-programme and the education process."

It is remarkable how Turing even describes approaches that have since evolved into established machine learning methods: evolution (genetic algorithms), punishments and rewards (reinforcement learning), and randomness (Monte Carlo tree search). He even forecasts the main issue with some forms of machine learning: opacity. "An important feature of a learning machine is that its teacher will often be very largely ignorant of quite what is going on inside, although he may still be able to some extent to predict his pupil's behaviour."

ENGLISH HERITAGE
ALAN TURING
1912~1954
Code-breaker and Pioneer of Computer Science was born here
© chrisdorney/stock.adobe.com

The evolution of a definition

The term ‘artificial intelligence’ was coined in 1956, at an event called the Dartmouth workshop. It was a gathering of the field’s founders; researchers who would later have a huge impact, including John McCarthy, Claude Shannon, Marvin Minsky, Herbert Simon, Allen Newell, Arthur Samuel, Ray Solomonoff, and W. S. McCulloch.

The simple and ambitious definition for artificial intelligence, included in the proposal for the workshop, is illuminating: “making a machine behave in ways that would be called intelligent if a human were so behaving”. These pioneers were making the assumption that “every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it”. This assumption turned out to be false, and led to unrealistic expectations and forecasts. Some 50 years later, McCarthy himself stated that “it was harder than we thought”.

Modern definitions of intelligence are of a distinctly different flavour: “Intelligence is the quality that enables an entity to function appropriately and with foresight in its environment” (**helloworld.cc/nilsson2009**). Some even speak of rationality: “doing the right thing, given what it knows” (**helloworld.cc/russell-norvig1995**).

> INTELLIGENCE IS THE QUALITY THAT ENABLES AN ENTITY TO FUNCTION APPROPRIATELY AND WITH FORESIGHT

Playing games: search

A lot of research in artificial intelligence has focused on games. Over the years, programs for playing draughts, backgammon, and many other games have reached competence levels that surpassed the best human players.

However, chess was the most prominent game, right from the start. Alan Turing and David Champernowne developed a basic algorithm called Turochamp for playing chess back in 1948. It took years for that algorithm to be implemented into a program, and Turing famously played a few games with human opponents executing the algorithm by hand. Claude Shannon wrote *Programming a Computer for Playing Chess* in 1950, in which he laid the foundation for many of the search techniques that would later be applied in games (**helloworld.cc/shannon1950**).

Search is the main approach for playing many of these games, systematically generating and evaluating positions and moves. That may sound trivial for a computer, but the number of combinations in non-trivial problems quickly explodes exponentially, and a brute-force enumeration of all the possible outcomes is impossible. Shannon estimated the number of different chess games to 10^{120}. In these vast search spaces, a lot of thought needs to go into evaluating search states, to guide the search effort and prune non-promising search paths.

In his paper, Shannon discussed the value of research in games, explaining that a solution “will act as a wedge in attacking other problems of a similar nature and of greater significance”. Indeed, search was the driving force behind many of the landmark achievements in the field: making plans and schedules, proving theorems with logic, solving algebraic problems, making inductions, and so on.

Deep Blue

In 1996, the chess world champion Garry Kasparov played against Deep Blue, a purpose-built IBM computer. Deep Blue became the first chess machine to ever win a game and, a year later, the first to win a match against a world champion under regular time controls. Kasparov’s defeat made the headlines and is considered a milestone in the history of AI.

The main driving force behind Deep Blue was search: it was able to generate and evaluate 200 million positions per second. The evaluation function was handcrafted by human experts, and the only form of learning was the system’s ability to fine-tune some of its parameters.

Modern chess-playing programs need to evaluate far fewer positions and do not require specialised hardware to vastly outperform human players. The last known win by a human against a top chess-playing machine was in 2005.

Deep Blue’s victory was part of an impressive string of achievements, but there were problems that seemed elementary and yet proved extremely hard to tackle. This was eloquently summarised in Don Knuth’s remark: “AI has by now succeeded in doing essentially everything that requires ‘thinking’ but has failed to do most of what people and animals do ‘without thinking’ — that, somehow, is much harder!” It is only very recently that AI has made breakthroughs in the latter class of problems, such as image and speech recognition, and this is the main reason it has become so prominent.

Watson

In 2011, Watson, a computer system built by IBM, competed against two human champions in a game of *Jeopardy!* The highly publicised match resulted in an impressive win for Watson, in a context that would traditionally have been considered extremely hard for a computer to tackle.

Watson is not really a computer system for playing *Jeopardy!*, though. It is a system that uses multiple different techniques to answer questions posed in natural

FURTHER READING

Artificial Intelligence by Michael Wooldridge **helloworld.cc/wooldridge2018**

Machine Learning for Humans by Vishal Maini and Samer Sabri **helloworld.cc/maini2017**

The Quest for Artificial Intelligence: A History of Ideas and Achievements by Nils Nilsson **helloworld.cc/nilsson2009**

Machines Who Think by Pamela McCorduck **helloworld.cc/mccorduck2004**

THE STRATEGY GAME GO WAS ALWAYS THE HOLY GRAIL OF GAME AI

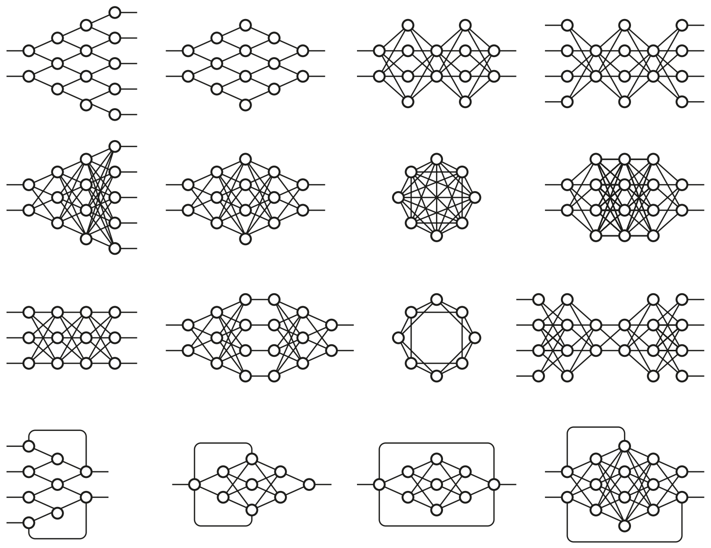
language. In order to answer a question, Watson generates multiple hypotheses and seeks to support them by drawing evidence from a body of sources. In other words, Watson is able to provide justification for its answers. There are many areas where Watson is now being applied; one of the most prominent involves assisting doctors with diagnosis and suggested treatment.

Watson's level of complexity is astonishing, and it would be impossible to develop such a system without some form of learning. Echoing Turing's comments about learning machines, Grady Booch, who was involved in building Watson, remarked that "building a cognitive system [like Watson] is fundamentally different than building a traditional software-intensive system of the past. We don't program them. We teach them."

Neural networks

A neural network receives input values and computes output values, which are influenced by a set of parameters called the weights. The function computed is a composition of simpler functions, represented by individual neurons. Building a neural network boils down to how these simpler functions are organised and composed (the network 'topology'), also taking the weights into account. A neural network can learn in the sense that its weights can be modified, swaying the output in more desirable directions.

Behind every recent breakthrough in artificial intelligence, you will find a neural network. Teaching a neural network of sufficient complexity requires a significant amount of training instances and computational power. Even though neural networks have been around for decades, it is only in the last few years that their potential has been realised, as the amount of available training data has skyrocketed and computing power, along with dedicated hardware, has become more readily available.



Neural networks are behind nearly all AI technology

AlphaGo

Go is an ancient strategy game for two players, who take turns placing black and white pieces (stones) on a 19x19 board.

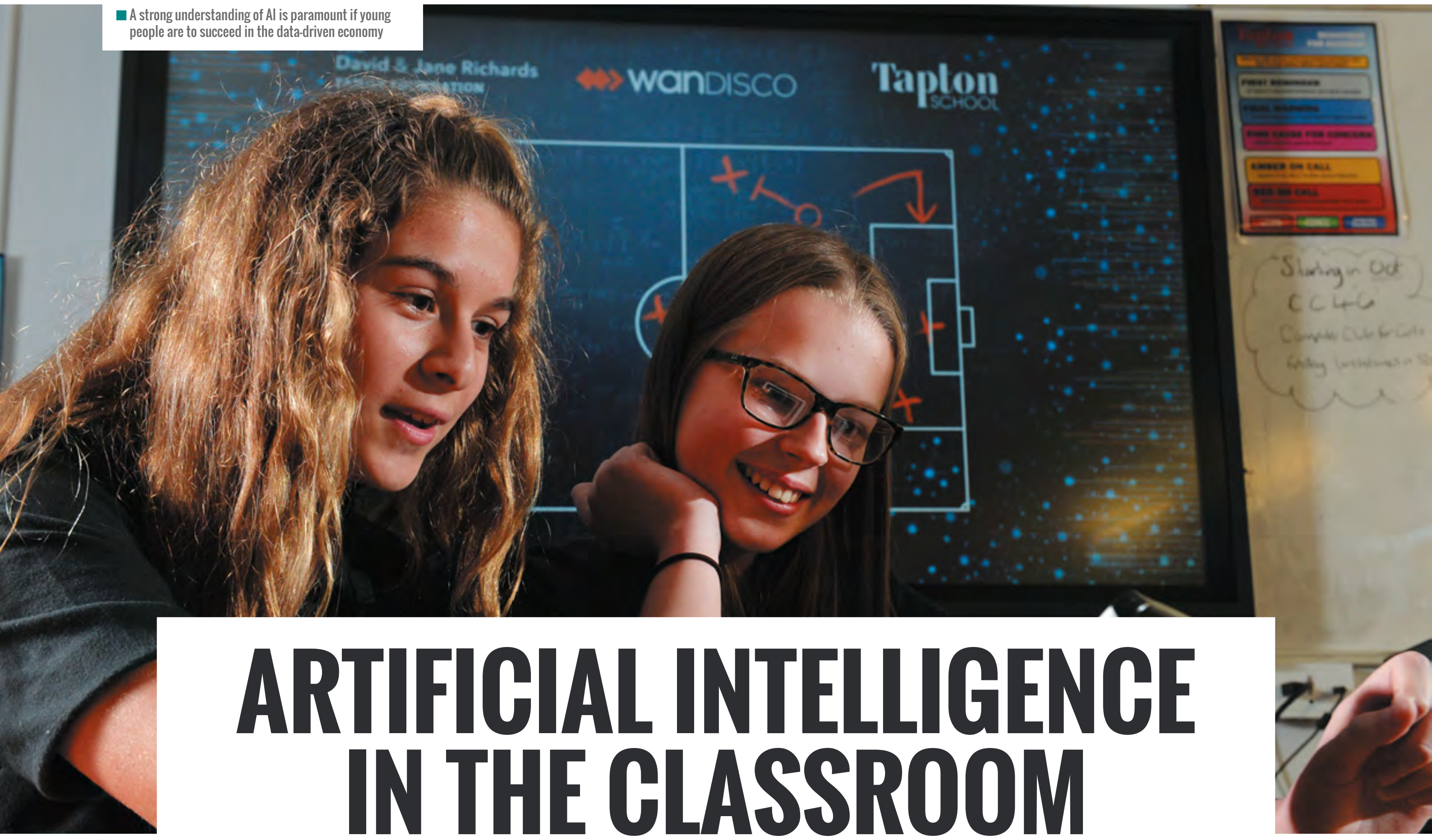
It is a notoriously difficult game for computers. The number of possible board positions is estimated at an astronomical 10^{170}. Traditional search techniques are pointless in such a vast space of possibilities, and it has proved very hard to develop functions that reliably evaluate positions in order to guide the search. Researchers estimated that it might take decades for machines to beat humans at Go, which was considered to be the holy grail of game AI.

Enter AlphaGo, a computer program by DeepMind. In 2017, AlphaGo beat Ke Jie, the world's top-ranked player at the time, following victories over other high-ranking professional players.

AlphaGo combines previously known methods in a novel way. It studies human games or uses self-play, in order to learn how to evaluate positions and moves. It uses neural networks to compute its evaluation functions and modify them while learning. It searches through the vast space of possible positions by taking random samples, instead of searching systematically.

This is such a promising generic approach that AlphaZero, a generalised version of the program, used only self-play to achieve a superhuman level of play in the games of chess, Shogi, and Go within 24 hours. This is a step closer to Turing's vision of a blank slate child-programme, endowed with the ability to learn. (HW)

GEORGE BOUKEAS
George is a former computing teacher and is now a Python engineer while figuring out what he wants to do when he grows up.



■ A strong understanding of AI is paramount if young people are to succeed in the data-driven economy

ARTIFICIAL INTELLIGENCE IN THE CLASSROOM

Emily Dreimann shares ideas for kick-starting discussions on artificial intelligence

There's little doubt that artificial intelligence (AI) has captured our collective imagination. TV series and films increasingly explore the implications of this technology, from family favourites such as *WALL-E* to the distinctly darker *Black Mirror*. Robotics companies, meanwhile, are transforming the more benign of these visions into reality: Hanson Robotics' lifelike robot Sophia, for example, has become a familiar sight on talk shows, and has even starred in music videos.

EMILY DREIMANN
Emily is a digital communications officer who previously worked at the David & Jane Richards Family Foundation (**@em_dreimann**).

At the same time, our understanding of how we personally interact with AI in our day-to-day lives, and how we can use it to our advantage, remains limited. Ask a class of 13-year-olds what they think of when they hear 'artificial intelligence', for example, and the answers tend towards a common theme: 'creepy'; 'sinister'; 'taking over the world'. How do we keep interest levels up while grounding AI in reality and preparing students for the workplaces of the future?

For young people to be able to lead and succeed in the data-driven economy, a strong understanding of this ever-evolving technology is paramount. To engage students with this topic, lessons should not only highlight the many forms that artificial intelligence can take in the real world, but also offer a tangible experience of and interactions with the technology. Here are just a few of the angles from which we can approach this topic, and suggestions for resources that can complement them.

AI in action

With the recent proliferation of smart speakers and virtual assistants, this technology can be a useful framework for an initial discussion around the key tenets of artificial intelligence. Most young people will have been exposed to these devices in some form; fewer, however, are likely to identify them as an example of AI. You could ask students:

- What does AI look like?
- What does it sound like?
- To what extent should it mirror human behaviour?

Google's Duplex AI assistant is a great example of the capabilities and potential of this technology. In a popular video of the

GET CREATIVE WITH DATA

AI and machine learning form a key part of *Get Creative With Data*, a data science course for students aged 11-14 from the David & Jane Richards Family Foundation (**@DJRichardsFF**).

Complete course materials are available for free to all state schools. For more information about introducing the course at your school, visit **djrff.org**.

assistant in action, we can hear it making calls to several businesses, sounding sufficiently humanlike in its interactions to fool the real humans on the other end of the line. This somewhat unnerving potential of AI is likely to hook students, while it remains grounded in reality as an aid and time saver, rather than a replacement, for humans.

Quick, Draw! is a great resource for highlighting to students another manifestation of AI. The game challenges the user to create a series of doodles, while a neural network attempts to guess what they are drawing. Coupled with its hands-on, accessible nature, this activity has the potential to engage even the most reluctant of students.

Not so intelligent?

At the same time, we need students to be critical in their appraisal of AI technology. The statement that 'Machine learning is written in Python; artificial intelligence is written in PowerPoint' is a great starting point for this discussion. You could ask students what they understand by this. The aim here is to draw out the idea that we can already see machine learning in action in industries across the globe, while AI arguably remains a theoretical concept. Has anyone yet created a truly intelligent machine?

There is a wealth of resources that we can draw on to assist students in forming their own opinions in this debate. The Turing test, for example, is an important concept for students to understand and remains a useful benchmark against which to measure the capabilities of AI technology. Encourage your students to read aloud some transcripts for entries to the Loebner Prize, a now defunct annual Turing test competition. Would any of these have convinced them they were speaking to a real human?

On the website AI Weirdness, meanwhile, research scientist Janelle Shane publishes the entertaining results of her experiments training neural networks on existing content across a range of topics, from cat names to knitting patterns. Taking Halloween costumes as an example, you could allow students to explore; with suggestions such as 'sentient stone' and 'a skunk in a moose suit', it should quickly become apparent to them that this technology has some way to go in capturing the uniquely human traits of creativity and humour.

© Anthony Brown/stock.adobe.com

Smart speakers can be a useful starting point for a discussion about AI

" AI ARGUABLY REMAINS A THEORETICAL CONCEPT, WHILE MACHINE LEARNING CAN ALREADY BE SEEN IN ACTION IN INDUSTRY

Branching out

AI and machine learning can also be ideal starting points for generating lively debate around other key topics in computing. An unplugged activity in which students create their own algorithm to guide a visitor from the school reception to their classroom is an ideal catalyst for a discussion about the differences between how people and machines make sense of instructions. We can highlight the ability of humans to apply common sense when determining a course of action as a strength that machines cannot emulate. This can encourage students to understand AI as a tool to complement us, rather than compete with us.

Meanwhile, developments in self-driving vehicle technology present a unique opportunity for students to explore ethics in the context of computer science. Moral Machine, developed by the Massachusetts Institute of Technology, is an interactive tool that asks the user to judge the most acceptable outcomes of a series of moral dilemmas faced by a self-driving car. By engaging with this modern take on the classic trolley problem, students develop a deeper and more personal understanding of the ethical challenges surrounding artificial intelligence.

These activities have been popular in schools I have worked with. Teachers have commented that pupils were "fully engaged" and found the topic "really interesting". By providing students with an interactive forum in which to explore AI, we have an excellent opportunity to support the next generation in confidently claiming their place in the modern world. (HW)

IMPACT OF TECHNOLOGY

Learners are constantly surrounded by technology, and new innovations impact every part of their lives. For them to become successful digital citizens or go on to develop new technologies themselves, learners need to understand the impact that technology has on individuals and on society at large. Technology may impact for better or worse, and in a wide range of ways, including health and well-being, careers, privacy, global politics, equality, and the environment.

Our approach to learning about the impact of technology begins with learners identifying technology around them and its direct impact on them as individuals, both at home and at school. Beyond this focus, they then start to explore the benefits and issues for society, specifically through the lens of moral, ethical, and legal frameworks, and the rights of individuals and organisations. Ultimately, they begin to think more from a global perspective, considering issues of equity and challenges facing the environment. Throughout their study, learners will encounter familiar examples and contexts, as well as considering the challenges presented by emerging technologies such as artificial intelligence and machine learning.

IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** impact of technology, in summary
- **What the research says:** contexts to inspire
- Morals and ethics in computer science
- What the law says about hacking
- Upgrade culture
- Air pollution project

IN SUMMARY

IMPACT OF TECHNOLOGY

Understand the impact of computer systems on individuals, organisations, and society

STAGE 1	STAGE 2
Identify and describe familiar examples of information technology	Identify tasks that are managed by computer systems
Identify information technology that can be used in more than one way	**Recognise that human decisions determine the use and impact of information technology**
Use information technology for different activities	Explain how the internet allows data to be shared globally
Explain why we use information technology and some of its benefits	**Explain the benefits of a given information technology system**
	Explain similarities and differences between digital and non-digital tools
	Consider ownership and copyright when selecting content

In the table below, you will find learning outcomes associated with the 'Impact of technology' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Analyse information sources to judge their credibility	Contrast the varying access, knowledge, and opportunities of different global communities in relation to information technology	Identify the challenges of legislating and enforcing the law in a global digital world
Explain the benefits of information technology for individuals and wider society	**Identify the positive and negative impact that computing has on the environment**	**Discuss the impact that emerging technologies will have on the workforce and economies**
Describe the potential harms of information technology upon society and individuals	Describe the sustainability challenges associated with digital technologies	Outline the data protection principles that developers and organisations must comply with
Recognise the need for laws related to the misuse of computers and data	**Outline the benefits and risks to individuals as a result of emerging technologies (for example AI)**	**Describe the laws that aim to protect individuals from malicious communication**
Describe the difference between legal, ethical, and moral behaviours	**Describe the rights that individuals have in relation to their personal data**	Evaluate the societal impacts of digital technologies, including how we work, communicate, learn, and entertain
Consider content copyright and credit sources of information	Outline laws that aim to protect individuals and organisations from digital crime	**Compare the positive and negative impacts of digital technologies on our physical and mental health**
Discuss uses of information technology that may be considered immoral or unethical	**Describe examples of situations where digital technologies introduce bias or reduce equity**	Identify factors that affect the accessibility, usability, and cultural relevance of digital technologies
Explain the importance of seeking diverse perspectives in evaluating and improving digital artefacts		**Discuss the ways in which technological products and tools cater for all members of a diverse population**

INSPIRING YOUNG PEOPLE WITH CONTEXTS THEY CARE ABOUT

STORY BY Oliver Quinlan

Many young people get into making things with computers through an early interest in the technology itself. There are many others who may not be as fascinated by technology for its own sake, but given the right context, can see it as a powerful tool to make a difference to the things they care about.

One such compelling context is health and well-being. We see just how many children care deeply about issues in this area every year from the projects entered to Coolest Projects, the Raspberry Pi Foundation event showcasing young people's digital projects.

Back in 2018, the Foundation carried out some research looking at the stories behind the projects children presented at the events in the UK and Ireland. One of the most highly regarded projects at the UK event was Be Healthy, an app to guide people's diets and health habits. The young creator of this app took a holistic view of health and created a single app that could encourage people to live healthy lives in a variety of ways. They focused very much on the design of the app, starting with the goals they wanted to achieve and then exploring how they would realise them with the technology.

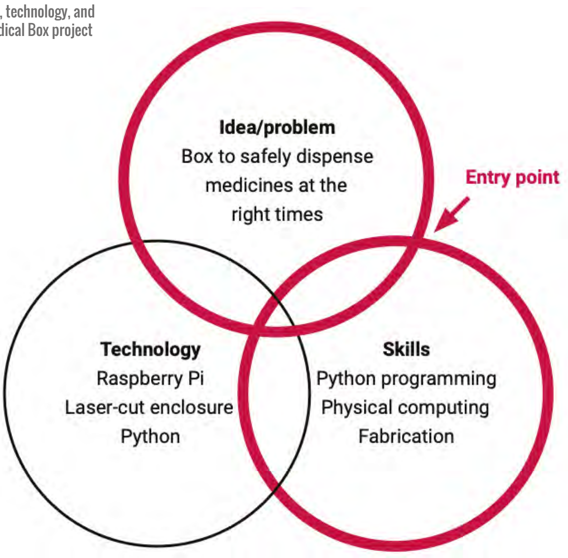
Another was Locking Medical Box, a physical computing project that supported people with mental health issues with managing their medication. The idea for this project came from the interest in health issues of one of the project team and drew on the experience of their parent, who was working in healthcare. The team had worked together before and met through a summer coding camp. For this project, they wanted to take skills they had learnt together and apply them to an area they all cared about. They combined a lot of thoughtful design work with both hardware and software skills to realise their idea as a working prototype.

Ideas, technology, and skills

From our research into these and many other projects, we put together a framework to help us understand the different ways in which young people approach making digital projects. We found that successful digital projects involve three key areas: ideas, technology, and skills.

What tends to happen is that young people start their projects with an emphasis on one of these three areas. Many start with the technology that they have access to and explore its potential uses. This is an important consideration,

The interaction of ideas, technology, and skills in the Locking Medical Box project



The Be Healthy app creator at Coolest Projects

as a lack of access to technology can be very limiting. This approach appeals to young people with a strong interest in technology for its own sake, and allows them to explore the possibilities of particular technologies.

Some young people start with skills they have learnt or want to learn, and create a project to apply these skills. In our research, we found that this approach was less common than the other two. It is a focus that we saw being taken by young people who already had quite a lot of experience with computing and digital making.

Other young people start with an idea in an area they care about and want to make a difference to. This approach can be a challenge for adults to facilitate, as it requires young people to figure out how they can execute ideas that are often quite ambitious, using technology they have access to and skills they can realistically acquire during the course of the project. It is a common tendency to be ambitious with ideas, but matching them to a comfortably challenging level of skill and the technology available to them can require some support.

Evidence from research into formal education shows that when computing lessons address contexts young people feel apply to them, such as health and well-being, it can help to motivate groups that are usually less engaged with computing, particularly girls (see the 'Further reading' box). This suggests that encouraging young people to approach digital making by focusing on the ideas that matter to them could be an effective way of opening up the activities to a wider range of students.

The Foundation is currently working together with Apps for Good (**helloworld.cc/appsforgood**) and the Behavioural Insights Team (**helloworld.cc/thebit**) to explore this approach to computing lessons in schools, as part of the RPF Gender Balance in Computing research project. For more information, and for examples of projects that you could share with young people, see the publication *How Children Make Digital Projects* at **helloworld.cc/projectsresearch**. (HW)

TIPS FOR TAKING AN 'IDEAS FIRST' APPROACH TO PROJECTS

- Make it clear to young people that the focus isn't on learning about technology for its own sake, but using technology to make a difference to things they care about.
- Share examples of projects linked to topics like health and well-being, and help young people see that they can use digital technology in contexts like this that they may not realise are possible.
- Set aside time away from the technology to talk to young people about the areas that interest them and the issues they care about; understanding this well will help you notice opportunities to build on their interests.
- Try to help learners break down their ambitious ideas into chunks or steps that might be more achievable with the skills, technology, and time available.
- Help learners work out what the minimum viable version of their idea is, so they can get to something functional quickly and feel a sense of achievement; they can always iterate and add to it later on.
- Be prepared for young people to want to do things you might not know how to do yourself. Support them to use forums (if age-appropriate) and online resources. You don't have to know everything, but you can help them figure out how to learn!

FURTHER READING

✓ Hulleman, C. S., & Harackiewicz, J. M. (2009). Promoting Interest and Performance in High School Science Classes. *Science, 326*(5958), 1410-1412. **helloworld.cc/hulleman2009**

✓ Kemp, P. E. J., Berry, M. G., & Wong, B. (2018). *The Roehampton Annual Computing Education Report.* University of Roehampton, London. **helloworld.cc/kemp2018**

FEATURE

HOW MORAL IS YOUR MACHINE?

Diane Dowling explores the moral and ethical dimension of computer science education

The A-level computer science specification of all English exam boards requires students aged 17–18 to have the ability to 'articulate the individual (moral), social (ethical), legal, and cultural opportunities and risks of digital technology'.

The terms 'ethics' and 'morals' are sometimes used interchangeably, as both refer to behaviour that can be labelled as 'right' or 'wrong'. Ethics may be guided or directed by codes of conduct in schools or workplaces, or by faith leaders, for those who practise a religion. Ethical guidance is provided to computing professionals by external bodies such as the British Computer Society, which sets the professional standards of competence, conduct, and ethical practice for computing in the United Kingdom.

On the other hand, morals are guided by our own principles and a sense of permitted behaviour. Charles Darwin maintained that "Of all the differences between man and the lower animals, the moral sense or conscience is by far the most important." It is a generally accepted view that as humans we all have:

- The ability to anticipate the consequences of our own actions
- The ability to make value judgements
- The ability to choose between alternative courses of action

Although we all have the capacity for moral behaviour, our individual moral code is not biologically determined, but arises as a result of human experience. The society in which we live influences our morals; for young people, they will be formed by the views of parents, teachers, and other people they interact with. Increasingly for most of us, this will include content consumed through the internet. The internet does not respect national borders; thoughts and ideas can be readily shared on social media, in chat rooms, and on forums. Such a wide sphere of influence can and will result in diverse views of what is right and wrong, even between members of the same household.

Some moral values are widely held by most societies, but there can be shades of grey in even the most widely held beliefs. 'Thou shalt not kill' is a tenet of many religions, and most people, when asked, will agree that killing another human being is wrong. However, across the globe, 56 countries retain the death penalty, and research shows that in these countries, the majority of the population agrees that the penalty is an appropriate punishment for those in society who commit the most serious crimes.

An interesting dilemma arises when we have to choose between two alternative courses of action, where both are morally reprehensible. An example of such a dilemma is the much-studied trolley problem. In this thought experiment, there is a runaway trolley. Ahead, on the tracks, there are five people tied up and unable to move; the trolley is heading straight for them. You are standing, some distance away, next to a lever. If you pull this lever, the trolley will switch to a different set of tracks. However, you notice that there is one person on the side track. You have two options:

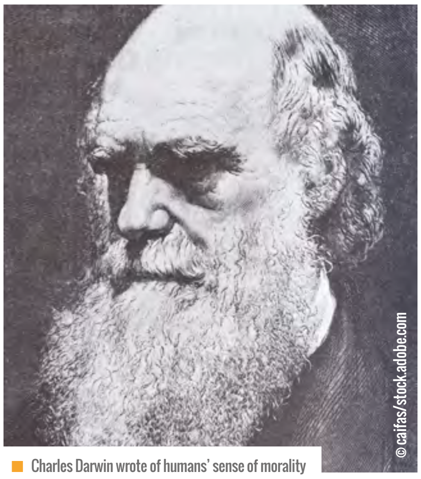
- Do nothing and allow the trolley to kill the five people on the main track
- Pull the lever, diverting the trolley onto the side track, where it will kill one person

The dilemma can be made more challenging by adapting the alternatives to include children or animals, or by varying age, gender, intelligence, or socioeconomic factors. Examples of how the trolley problem can be used in the classroom were given by Marc Scott on page 86 of Hello World issue 12 (**helloworld.cc/12**).

for making this impossible choice?

An algorithm may ensure that consistently reliable decisions are made, but whose morals will determine the way that autonomous machines are programmed? Would you rather a human could override a machine-made decision, or that rules were absolute and consistently applied? Machine learning muddies the debate still further. Neural networks used for this purpose are difficult to analyse in order to determine why a decision was produced, so there is a serious issue of accountability.

The Massachusetts Institute of Technology has created a website — **moralmachine.net** — that is collecting data to help researchers by providing a platform for "building a crowd-sourced picture of human opinion on how machines should make decisions when faced with moral dilemmas, and crowd-sourcing assembly and discussion of


© caifas/stock.adobe.com

Charles Darwin wrote of humans' sense of morality

“ DELEGATING MORAL DECISION-MAKING TO MACHINES CAN BE FASCINATING OR FRIGHTENING, DEPENDING ON OUTLOOK

Make the link

The moral conundrum is interesting, but how does any of this relate to computer science? The fact is that many of today's computer science students will become the software engineers of the future, and a large number will be faced with the task of designing and writing code for artificial intelligence applications such as self-driving cars. These programs will have to make complex, autonomous decisions that are often a matter of life or death.

The prospect of delegating moral decision-making to machines can be fascinating or frightening, depending on your outlook. The trolley problem can be abstracted to encompass many moral dilemmas. Sometimes, unprecedented situations such as the coronavirus pandemic will throw up new dilemmas. In hospitals worldwide, medics were forced to prioritise their patients. Who gets a ventilator when there is insufficient supply for all those who need one? Would it be better if a machine had the responsibility

potential scenarios of moral consequence". Projects such as this will allow researchers to gain a better understanding of the choices that humans make.

In guiding young people through the moral maze, there are many topics that can be discussed in the classroom to promote lively discussion. Facilitating a debate in which students propose arguments for both sides will allow a wide range of views to be shared and discussed. However, there are issues that need careful consideration; for example, when discussing autonomous vehicles, a young person with a friend or family member who has been involved in a traffic accident might find this a very hard topic to engage with.

Life-and-death choices are at the extreme end of the decision dilemma. There are many less contentious areas that could be discussed. In her excellent book, *Hello World: Being Human in the Age of Algorithms*, mathematician Hannah Fry introduces a range of topics, from medicine to law, where algorithms are already being used to automate decision-making. For example, in some US states, an algorithm that uses data about a defendant to estimate their likelihood of committing a future crime is deployed to recommend whether

© Goodpics/stock.adobe.com

Surveillance and privacy are hotly debated topics in ethics someone awaiting trial should be granted bail. If you search for more information on this, you will quickly find some fascinating examples of bias in the data.

Artificial intelligence and automated decision-making are not the only topics in which morals play an important role. The use of big data, and the ability of organisations and government to analyse personal information, are also worth discussing.

The power of the state to monitor behaviour is always contentious. Advances in facial recognition technology are enabling some regimes to monitor and track their citizens, putting in doubt the principle of informed consent. Since December 2019, all mobile phone users in China registering new SIM cards must submit to facial recognition scans, giving rise to suspicion of mass state surveillance.

Such an initiative would previously have been widely vilified by more democratic societies but, since coronavirus has swept the globe, many governments are now deploying tracking apps. In the UK, the government released an app to automatically collect details of those we had been in close contact with, to help control the spread of the virus. Many will applaud such initiatives, but defenders of civil liberties and the right to privacy will be dismayed by such developments; their moral code would not condone such a measure, even for the greater good.

The use of social media and other online platforms is another area that can facilitate lively debate. How much freedom should people have to express a viewpoint? Where is the line between what is allowed and what should be banned? In the 2019 UK General Election, many female candidates said that they felt unsafe. In research funded by Joseph Rowntree Reform Trust, analysis of 139,564 tweets sent on a single day in November 2019, which either replied to or otherwise mentioned any of the 2503 election candidates who used Twitter, found that 23,039 (16.5 percent) of the total were abusive.

> “ INTEGRATING THE STUDY OF MORALS AND ETHICS WILL RELATE REAL-WORLD ISSUES TO MORE THEORETICAL TOPICS

Of greater relevance to teenagers might be the death of Molly Russell, a 14-year-old girl who took her own life in 2017. Her Instagram account contained distressing material about self-harm and suicide. Molly's father claimed the algorithms used by some online platforms push similar content towards you, based on what you had been previously looking at.

However, there are also stories of social media being used for collective change. In 2017, actor Alyssa Milano encouraged people to say ‘me too’ if they had experienced sexual harassment or assault, and the #MeToo hashtag quickly swept the globe and empowered victims to speak out. Social media also gives a voice to many who live in less liberal societies. In Hong Kong, activists have been able to use social media to communicate and organise large-scale demonstrations against what they saw as an attempt by the Chinese government to undermine the region's autonomy and civil liberties.

Plan for morals and ethics

Integrating the study of morals and ethics into your scheme of work for computer science will provide the opportunity to relate real-world issues to more theoretical topics, and to make them relevant to the world we live in. Many of the topics introduced in this article are emotionally challenging, and teachers may feel uncomfortable introducing them into the classroom. However, many of these issues are relevant and important for older learners who are nearing adulthood. Teachers have a unique place in the lives of young people and an important role in steering and guiding their moral development. (HW)

Sometimes, social media can inspire positive change

DIANE DOWLING
Diane is a learning manager at the Raspberry Pi Foundation, where she works on the Isaac Computer Science platform. In her spare time, she is a trustee of a national charity that runs robotics events for sixth-formers.

THE THIN BLUE PIXELATED LINE

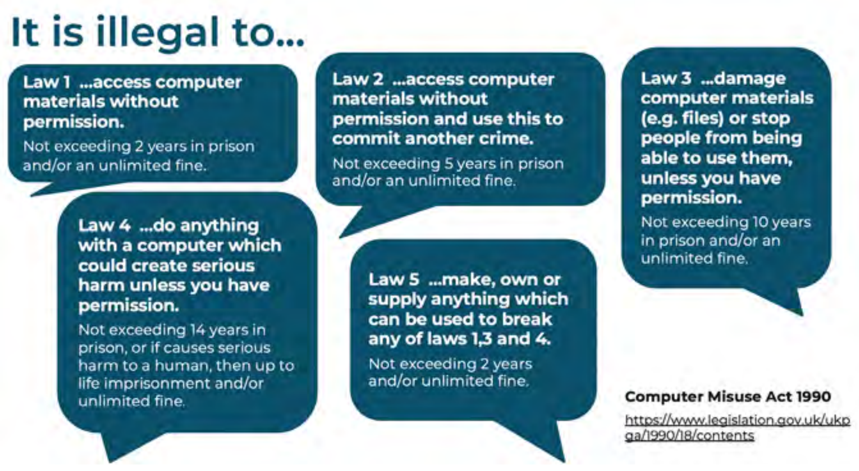
Jon Chippindall and **Alan Merrett** discuss what the law says about hacking and the resources you can use to educate your learners

Mention 'hacking' or 'hackers' to your class and you might spot a few smiles or exchanged glances. There might even be confessions of "I managed to hack so-and-so's Roblox account once." Pupils' eagerness to identify as hackers might be somewhat explained by the romanticisation of hacking in popular culture. But let's be absolutely clear: without express permission, unauthorised access to any computer system is illegal, and our pupils need to know this so they don't find themselves on the wrong side of the law. In this article, we'll unpick a little about what the law says around hacking and other cybercrimes, and share some resources you can use to educate your pupils. Although this article focuses on UK law, its learnings should still be relevant for all.

"It's OK, I only teach primary"

You might question whether we need to be covering the legalities of hacking in primary schools, as surely these pupils are too young to be at risk of offending — but you'd be surprised. The *Pathways Into Cyber Crime* report from the UK's National Crime Agency (NCA) highlighted that 61 percent of hackers start hacking before the age of 16, and many can trace their pathways back to discussions in game-modding forums at the top end of primary-school age (**helloworld.cc/ncareport**). Looking at the makeup of many digital literacy curricula, perhaps we've spent too much time on teaching how to protect against cybercriminals at the expense of reminding pupils not to become cybercriminals themselves! What's more, the advent of new off-the-shelf hacking tools has lowered the bar of the technical knowledge required to undertake cyberattacks, so it's more important than ever that we educate pupils from a young age on what is and isn't OK to do with computers.

There is also a real positive opportunity here to promote cybersecurity career choices and strike an appropriate balance of stick and carrot. Pupils need to know what is illegal, but they also need to know that if they have an interest in the computer science behind hacking, there is an ever-growing world of jobs available to them when they're a bit older. Many offenders interviewed by the NCA and police officers were motivated not through malicious intent, but by genuine curiosity, and by the satisfaction to be gained from solving the complex technical challenge of the hack. If we can get these pupils on the right track, they'll be a huge asset to any organisation that employs them to defend their digital interests. So, what does the law actually say?



It is illegal to...

Law 1 ...access computer materials without permission.
Not exceeding 2 years in prison and/or an unlimited fine.

Law 2 ...access computer materials without permission and use this to commit another crime.
Not exceeding 5 years in prison and/or an unlimited fine.

Law 3 ...damage computer materials (e.g. files) or stop people from being able to use them, unless you have permission.
Not exceeding 10 years in prison and/or an unlimited fine.

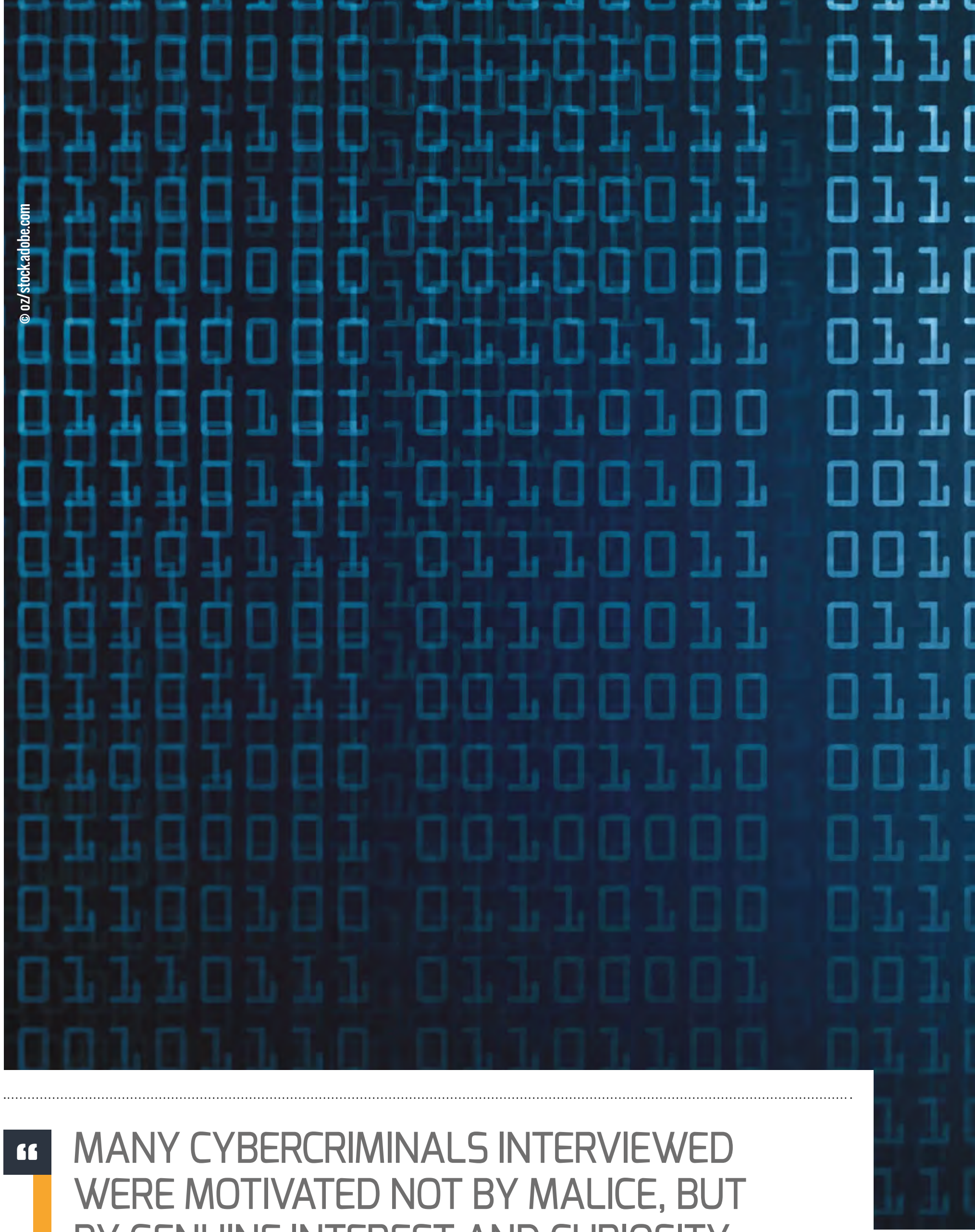
Law 4 ...do anything with a computer which could create serious harm unless you have permission.
Not exceeding 14 years in prison, or if causes serious harm to a human, then up to life imprisonment and/or unlimited fine.

Law 5 ...make, own or supply anything which can be used to break any of laws 1,3 and 4.
Not exceeding 2 years and/or unlimited fine.

Computer Misuse Act 1990
https://www.legislation.gov.uk/ukpga/1990/18/contents

Figure 1 Child-friendly translations of the laws constituting the UK's Computer Misuse Act 1990

>



FURTHER RESOURCES

barefootcomputing.org/cyber: all the activities mentioned in the article and more can be downloaded here.

cyberchoices@nca.gov.uk or **NCCUprevent@nca.gov.uk**: email addresses for additional cybercrime-related support in the UK.

What the law says

In the UK, the Computer Misuse Act 1990 (**helloworld.cc/cma1990**) sets out what constitutes illegal activity with a computer. There are five elements of the law, which were translated into child-friendly explanations as part of Barefoot's Be Cyber Smart resources; these are shown in **Figure 1**. This figure also shows details of the sentences people can receive if convicted of each element, including imprisonment and fines.

To bring the Computer Misuse Act to life, let's look at a selection of real prosecutions to illustrate the elements that make up the law. The following case information was taken from a record of convictions available at **helloworld.cc/cmacases**. This record could be explored with older pupils to deepen their understanding of when and how the law is broken, but please be aware that some cases include crimes of a sexual nature that are inappropriate for students.

Case 1

A 22-year-old student created software capable of harvesting names and passwords for various online services. They deployed the software to gather these credentials so that they could then access the services for free. They were imprisoned for six months after they were found guilty of creating the software to harvest login credentials (**breaking law 5**) and accessing the services without permission (**breaking law 1**).

> MANY CYBERCRIMINALS INTERVIEWED WERE MOTIVATED NOT BY MALICE, BUT BY GENUINE INTEREST AND CURIOSITY

Case 2

A disgruntled Jet2 employee launched a revenge attack that shut down Jet2's booking system and accessed the CEO's email. Recovery cost the company £165,000 (about $188,000). The person was convicted of accessing Jet2's files without permission and subsequently damaging them (**breaking laws 1 and 3**). They were sentenced to ten months in prison and their laptop was destroyed.

Case 3

A student hacked into social media and gaming accounts using a program they had created, and then sold the personal information from them (**breaking laws 1 and 4**). They were sentenced to four months' imprisonment, suspended for one year.

From these three cases alone, we can see the far-reaching impact of cybercrimes. Here, the victims include those whose online services were being used without their permission, and social media users whose personal information was sold without their knowledge — potentially leading to further crimes such as identity theft. In their interviews, the NCA learnt that cybercriminals often see their crimes as victimless. With large-scale hacks, though, the number of victims can vastly surpass those affected by traditional crimes, as demonstrated in these case studies. Most large businesses now employ cybersecurity teams to prevent hacks by regularly testing their organisation's defences. These penetration testers are just one example of the careers available in cybersecurity.

By introducing pupils to the Computer Misuse Act 1990 (or your country's equivalent) and case studies such as these, we can lead discussions to educate pupils on what constitutes the illegal use of computers, the impact of these crimes, and the sentences perpetrators can receive. Here is a selection of questions to lead a discussion with pupils:

- How was the law broken?
- What specific part of the law was broken?
- Who are the victims? How are they affected?
- What might the punishment be?

Taking this one step further, the free Barefoot You're the Jury resources, which can be downloaded for free at **barefootcomputing.org/cyber** after a quick registration, suggest turning classrooms into courtrooms and putting pupils in the roles of defendant, barrister, and jury members. Pupils then hear a number of cybercrime cases and for each, consider whether the law has been broken, who the victims are, and what the punishment might be. The resources even include templates to create a barrister's wig for pupils and a judge's wig for the teacher!

From the same set of resources, You're the Cyber Security Expert brings us back to the positive opportunities of the topic, namely raising awareness of cybersecurity careers. It does this by giving pupils a taste of life in this field as they learn what a brute-force hack is and, importantly, what strategies we could deploy to guard against it. So, for those pupils who exchanged a knowing glance at the mention of hacking, let's harness that curiosity and raise their awareness of the rewarding careers that pursue this interest legally. (HW)

JON CHIPPINDALL & ALAN MERRETT

Jon (pictured) is the Barefoot director and the computing lead at Crumpsall Lane Primary School in Manchester, UK. He also leads the computing PGCE at The University of Manchester. Alan is a senior officer in the Cyber Prevent Team at the UK's National Crime Agency. He has 34 years of law enforcement experience, including overt and covert roles with HM Customs and a team leader role at the UK's Interpol Desk and Fugitives Unit.

DOES UPGRADE CULTURE NEED AN UPGRADE?

Mac Bowley questions our habit of swiftly replacing devices with newer versions, with some interesting discussion points for students

Technology is more embedded in our lives than ever before, and most of us now carry a computer in our pocket everywhere we go. On top of that, the length of time for which we use each individual piece of technology has rapidly decreased. This is what's referred to as upgrade culture, a cycle that sees many of us replacing our most trusted devices every two to three years with the latest products offered by tech giants such as Apple and Samsung.

How we got to this point is hard to determine, and there does not seem to be a single root cause of upgrade culture. This is why I want to start a conversation about it, so we can challenge our current perspectives and establish fact-based attitudes. I think it's time that we, as individuals and as a society, examined our relationship with new technology.

What is the natural lifespan of a device?

Digital technology is still so new that there is really no benchmark for how long digital devices should last. This means that the decision-making power has by default landed in the hands of device manufacturers and mobile network carriers, and for their profit margins, a two- or three-year life cycle of devices is beneficial.

Where do you, as a consumer, see your role in this process? Is it wrong to want to upgrade your phone after two or three years of constant use? Should phone companies slow their development, and would this hinder innovation? And, if you really need to upgrade, is there a better use for your old device than living in a drawer? These questions defy simple answers, but your students should be aware of their role in this process as consumers, so this is a great area for in-class discussion.

How does this affect the environment?

As with all our behaviours as consumers, the impact that upgrade culture has on the environment is an important concern. Environmental issues and climate change aren't anything new, but they're currently at the forefront of the global conversation, and with good reason.

There are a number of issues around the manufacture of our mobile devices, such as the large amounts of energy required. Here, though, I would like to focus on another

Is it wrong to want to upgrade your phone after two or three years of constant use?

© Kirill Gorlov/stock.adobe.com

■ A common trope of mobile phone adverts is the overwrought comparison of your current device with a newly launched version

aspect of the environmental impact of device production: the materials that are used to create some of the tiny components that form our technological best friends.

Some components of your phone cannot be created without using rare chemical elements, such as europium and dysprosium. (In fact, there are 83 stable non-radioactive elements in the periodic table, and 70 of them are used in some capacity in your phone; see **helloworld.cc/phoneelements**). Upgrade culture means there is high demand for these materials, and deposits are becoming more and more depleted. If you're hoping there are renewable alternatives, you'll be disappointed: a study by researchers working at Yale University found that there are currently no alternative materials that are as effective (**helloworld.cc/graedel2013**).

Then there's the issue of how the materials are mined. The market trading these materials is highly competitive, and more often than not, manufacturers buy from the companies that offer the lowest prices. To maintain their profit margins, these companies have to extract as much material as possible, as cheaply as they can. As you can imagine, this leads to mining practices that are less than ethical or environmentally friendly. As many of the mines are located in distant areas of developing countries, these problems may feel remote to you, but they affect a lot of people and are a direct result of the market we are creating by upgrading our devices so frequently.

“ WE HAVE LET DEVICE MANUFACTURERS AND NETWORK CARRIERS TELL US HOW LONG OUR DIGITAL DEVICES SHOULD LAST

Many of us agree that we need to do what we can to counteract climate change, and that to achieve anything meaningful, we have to start looking at the way we live our lives. This includes questioning how we use technology. It will be through discussion and opinion-gathering that we can start to make more informed decisions — both as individuals and as a society.

The obsolescence question

You probably also have that one friend/colleague/family member who swears by their five-year-old mobile phone and scoffs

>

The length of time for which we use individual pieces of technology has declined rapidly

at the prices of the newest models. These people are often labelled as sticklers who are afraid to join the modern age, but is there another way of seeing them? The truth is, if you've bought a phone in the last five years, then — barring major accidents — it will most likely still function, and be just as effective as it was when it came out of the box. So why are so many consumers upgrading to new devices every two or three years?

There isn't a single reason, but I believe marketing departments should shoulder much of the responsibility. Using marketing strategies, device manufacturers and mobile network carriers purposefully encourage us to view the phones we currently own in a negative light. A common trope of mobile phone adverts is the overwrought comparison of your current device with a newly launched version. Thus, with each passing day after a new model is released, our opinion of our device worsens, even if only on a subconscious level.

This marketing strategy is related to a business practice called planned obsolescence, which sees manufacturers purposefully limiting the durability of their products in order to sell more units. An early example of planned obsolescence is the light bulb, invented by the Edison company. It was relatively simple for the company to create a light bulb that would last 2500 hours, but it took years and a coalition of manufacturers to make a version that reliably broke after 1000 hours (**helloworld.cc/lightbulb**). We're all aware that the light bulb revolutionised many aspects of life, but it turns out it also had a big influence on consumer habits and on what we see as acceptable practices by technology companies.

Manufacturers often purposefully limit the durability of their products in order to sell more units

> WE ARE WIDENING THE DIGITAL DIVIDE BY PLACING MORE VALUE ON NEW TECHNOLOGY THAN IS WARRANTED

The widening digital divide

The final aspect of the impact of upgrade culture that I want to examine relates to the digital divide. This term describes the societal gap between the people with access to, and competence with, the latest technology, and the people without these privileges. To be able to upgrade, say, your mobile phone to the latest model every two years, you either need a great degree of financial freedom, or you need to tie yourself to a 24-month contract that may not be easily within your means. As a society, we revere the latest technology and hold people with access to it in high regard. What does this say to people who do not have this access?

Inadvertently, we are widening the digital divide by placing more value on new technology than is warranted. Innovation is exciting, and commercial success is celebrated — but do you ever stop and ask who really benefits from this? Is your new phone really that much better than the old one, or could it be that you're mostly just basking in the social rewards of having the newest bit of kit?

What do you think? Time for you to discuss with your students! Here are some discussion starters to use with them:

- When you are in charge of buying your own phone, what can you do to make the device last longer than the usual two- to three-year upgrade cycle?
- Do you think upgrade culture is something that should be addressed by mobile phone manufacturers and providers, or is it caused by our own consumption habits?
- How might we address upgrade culture? Is it a problem that needs addressing?

MAC BOWLEY
Mac is a learning manager at the Raspberry Pi Foundation. When he isn't teaching, you can usually find him tinkering with his latest project (**@Mac_Bowley**).

Upgrade culture is one of the topics for which we offer you a discussion forum on our free online *Impact of Technology* course (**helloworld.cc/impactoftech**). The course, designed for educators, also covers how to facilitate classroom discussions about these topics — sign up today to take part for free! (HW)

WHAT ABOUT RASPBERRY PI TECHNOLOGY?

Obviously, this article wouldn't be complete if we didn't share our perspective as a technology company. So here's Raspberry Pi Trading CEO Eben Upton:

ON OUR HARDWARE AND SOFTWARE

"Raspberry Pi tries very hard to avoid obsoleting older products. Obviously the latest Raspberry Pi 4 (**helloworld.cc/pi4**) runs much faster than Raspberry Pi 1 (something like 40 times faster), but a Raspberry Pi OS (**helloworld.cc/rpiOS**) image we release today will run on the very earliest Raspberry Pi prototypes from the summer of 2011. Cutting customers off from software support after a couple of years is unethical, and bad for business in the long term: fool me once, shame on you; fool me twice, shame on me. The best companies respect their customers' investment in their platforms, even if that investment happened far in the past.

"What's even more unusual about Raspberry Pi is that we aim to keep our products available for a long period of time. So not only can you run a 2020 software build on a 2014 Raspberry Pi 1B+; you can actually buy a brand-new 1B+ to run it on (**helloworld.cc/pi1b**)."

ON THE ENVIRONMENTAL IMPACT OF OUR HARDWARE

"We're constantly working to reduce the environmental footprint of Raspberry Pi. If you look at the USB connectors on Raspberry Pi 4, you'll see a chunky black component. This is the reservoir capacitor, which prevents the 5V rail from dropping too far when a new USB device is plugged in. By using a polymer electrolytic capacitor from our friends at Panasonic, we've been able to avoid the use of tantalum.

"When we launched the official USB-C power supply for Raspberry Pi 4 (**helloworld.cc/piUSBC**), one or two people on Twitter asked if we could eliminate the single-use plastic bag that surrounded the cable and plug assembly inside the box. Working with our partners at Kuantech, we found that we could easily do this for the white supplies, but not for the black ones. Why? Because when the box vibrates in transit, the plug scuffs against the case; this is visible on the black plastic, but not on the white. So for now, if you want to eliminate single-use plastics, buy a white supply. In the meantime, we'll be working to find a way (probably involving cunning origami) to eliminate plastic from the black supply."

PROTECTING CHILDREN FROM BREATHING HAZARDOUS AIR

James Abela shares how his computer science students solved the very real problem of predicting air pollution and looks at the classroom environment that made this possible

In 2018, Indonesia burned approximately 529,000 hectares of land. That's an area more than three times the size of Greater London, or almost the size of Brunei. With so much forest being burned, the whole region felt the effects of the pollution. Schools frequently had to ban outdoor play and PE lessons, and on some days, schools were closed completely. Many schools in the region had an on-site CO_2 detector to reveal when pollution was bad, but by the time the message could get out, children had already been breathing in the polluted air for several minutes.

My Year 12 students (aged 16–17) followed the news and weather forecasts intently, and we all started to see how the winds from Singapore and Sumatra were sending pollution to us in Kuala Lumpur. We also realised that if we had measurements from around the city, we might have some visibility as to when pollution was likely to affect our school.

Arduino sensor used for detecting pollution

Making room for student-led projects

I've always encouraged my students to do their own projects, because it gives programming tasks meaning and creates something they can be genuinely proud of. The other benefit is that it gives them something to talk about in university essays and interviews, especially as they often need to do extensive research to solve the problems central to their projects.

This project was much more than this: it was a genuine passion project in every sense of the word. Three of my students approached me with the idea of tracking CO_2 to give schools a better idea of when there was pollution and which way it was going. They'd had some experience of using Raspberry Pi computers, and knew that it was possible to use them to make weather stations, and that the latest versions had wireless LAN capability that they could use. I agreed to support them during allocated programming time, and to help them reach out to other schools.

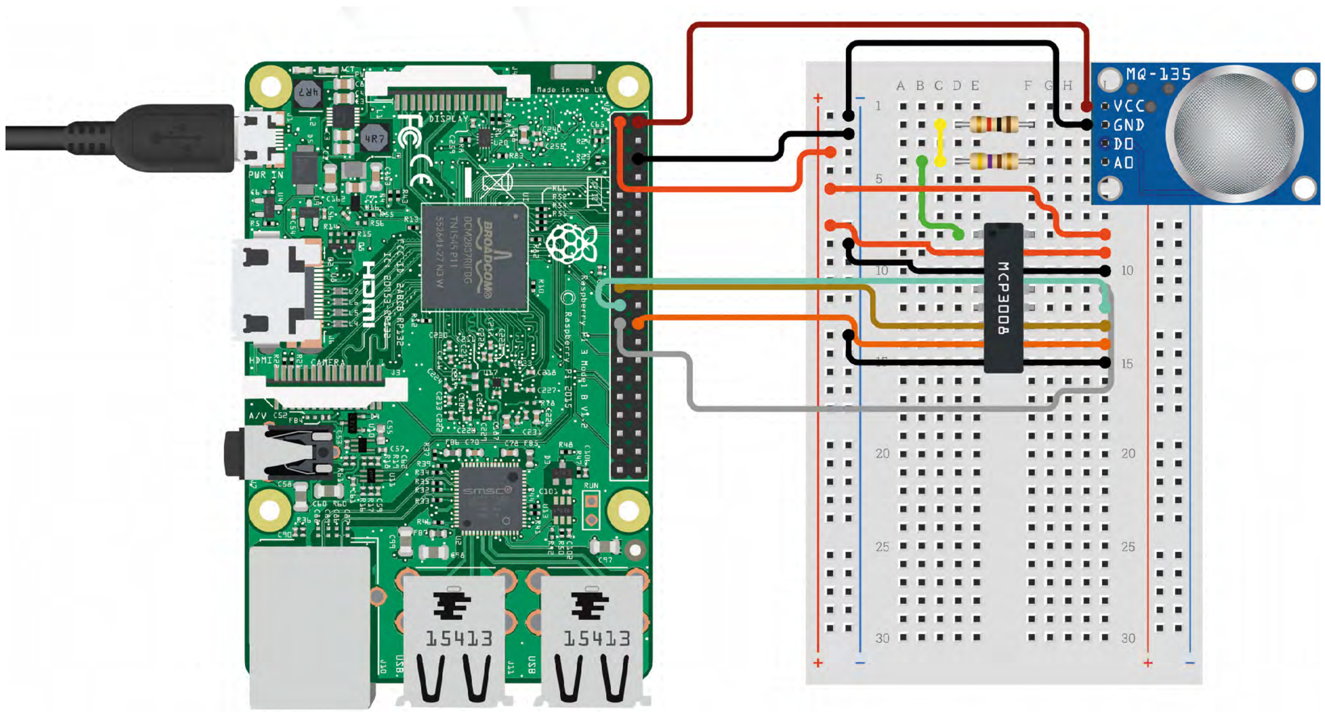
I was able to offer students support with this project because I flip quite a lot of the theory in my class. Flipped learning is a teaching approach in which some direct instruction, for example reading articles or watching specific videos, is completed at home. This enables more of our class time to be used to answer questions, work through higher-order tasks, or do group work, and it creates more supervised coding time.

I initially started doing this because when I set coding challenges for homework, I often had students who confessed they had spent all night trying to solve it, only for me to glance at the code and notice a missing colon or indentation issue. I began flipping the less difficult theory for students to do as homework, to create more programming time in class where we could resolve issues more quickly. This then evolved into a system in which students could work much more at their own pace, and eventually led to a point at which older students could, in effect, learn through their own projects, such as the pollution monitor.

Building the pollution monitor

The students started by looking at existing weather station projects — for example, there is an excellent tutorial at **helloworld.cc/weatherstation**. Students discovered that wind data is relatively easy to get over a large area, but the key component would be something to measure CO_2.

We found a sensor (the CCS811 sensor module) on a Malaysian site called Lazada. It was designed to work with an Arduino, and so we connected our Raspberry Pi via its USB port to an Arduino, and so to the CO_2 sensor. You could also order a more accurate sensor directly from the Arduino store (the MG-811 sensor) or order a variety of sensors from eBay, such



Circuit design of the CO_2 sensor using just Raspberry Pi; *designed on circuito.io*

as the MQ-135 hazardous gas sensor. We then used the *Get started with Arduino* guide (**helloworld.cc/getstartedarduino**) to help us connect the two together. The advantage for us was that the CO_2 sensor module we bought was designed to interface with the Arduino, so it was easier to install. It is also possible to connect Raspberry Pi directly to such a sensor using a breadboard, an analogue-to-digital converter (MCP3008 will work), a 1K resistor, and a 470 ohm resistor (see the circuit design image above).

We were very pleased to see that data started to come through showing us the CO_2 levels. Our plan was to run the Raspberry Pis headless and export this data to Google Sheets. We found an excellent way to do this in Python using the Google Sheets API (**helloworld.cc/sheetsapi**). This meant that our spreadsheet was automatically loaded with real data, and from there we could make a visualisation to show the CO_2 data as it was being generated. We also contacted other schools around Kuala Lumpur to see if they would be interested in putting a device on their roof, and most were interested in the idea.

Beaten to the punch

We were not the only ones with such an idea, and a company called IQAir began selling the AirVisual Pro around this time, which did almost exactly what we hoped to do, and did it incredibly well. Schools were already very receptive to the idea and quickly invested in the technology. It is still very impressive to think that three Year 12 students came up with an idea that solved the very real concern of pollution visibility and were only fractionally behind a commercial solution.

This project really helped these students to decide whether they enjoyed the hardware side of computing, and solving real-world issues encouraged them to see computing as a practical subject. This is a message that has resonated with other students, and we've since doubled the number of students taking A-level computer science. Since doing this project, I've encouraged students to take on the Extended Project Qualification (**helloworld.cc/extendedproject**). This will give them time to explore concepts fully and allow them to put their programming to good use, tackling problems that interest them and that the world needs solving. (HW)

“ IT WAS A GENUINE PASSION PROJECT IN EVERY SENSE

JAMES ABELA
James is the head of computing at Garden International School in Kuala Lumpur, Malaysia. He is an RPi Certified Educator, founder of the South East Asian Computer Science Teachers Association, and author of *The Gamified Classroom* (**@eslweb**).

DESIGN AND DEVELOPMENT

Whenever learners develop a digital artefact, whether it be a program, video, database, or something else, they will typically engage in some design and development activities. These activities incorporate analysis, planning, implementation, testing, and evaluation. Throughout their journey in computing, learners should apply and hone these skills using relevant tools and techniques. Most learners will have some previous experience of design processes from other subjects, but they may not necessarily associate them with computing.

In the early stages of their learning, there is a focus on learners planning and communicating their ideas and discovering the value in planning before they put those ideas into practice. They will also learn to express the key requirements of a task and to provide feedback on their own or others' solutions. As they progress, they will encounter more formalised approaches to designing, testing, and evaluating their products. Increasingly, the responsibility for analysis, user research, and establishing success criteria for a particular project will shift to the learners themselves, until they are able to independently design and develop significantly complex digital artefacts that meet the needs of their audience.

IN THIS SECTION, YOU WILL FIND:

IN SUMMARY

DESIGN AND DEVELOPMENT

Understand the steps involved in analysing, planning, creating, and evaluating computing artefacts

STAGE 1	STAGE 2
Outline the broad requirements of a task	Explain the key requirements a digital artefact has to fulfil to meet the needs of its audience
Experiment with different tools and approaches to understand what is possible before building a solution	**Explain the limits of what is possible with available tools, time, and understanding**
Use planning templates to sketch and describe solutions before building them	Describe the benefits of planning before creating a project
Follow a plan to create a solution	**Create structured plans for solutions using templates (flowcharts, storyboards, diagrams)**
Choose appropriate media from a limited selection	Create a first draft or prototype of a solution, including key features
Gather and provide feedback on finished artefacts	**Iterate and improve a solution incorporating suitable media**
Check that a solution meets the task requirements	Test and gather feedback on a solution throughout its development
	Evaluate the success of a solution for the task

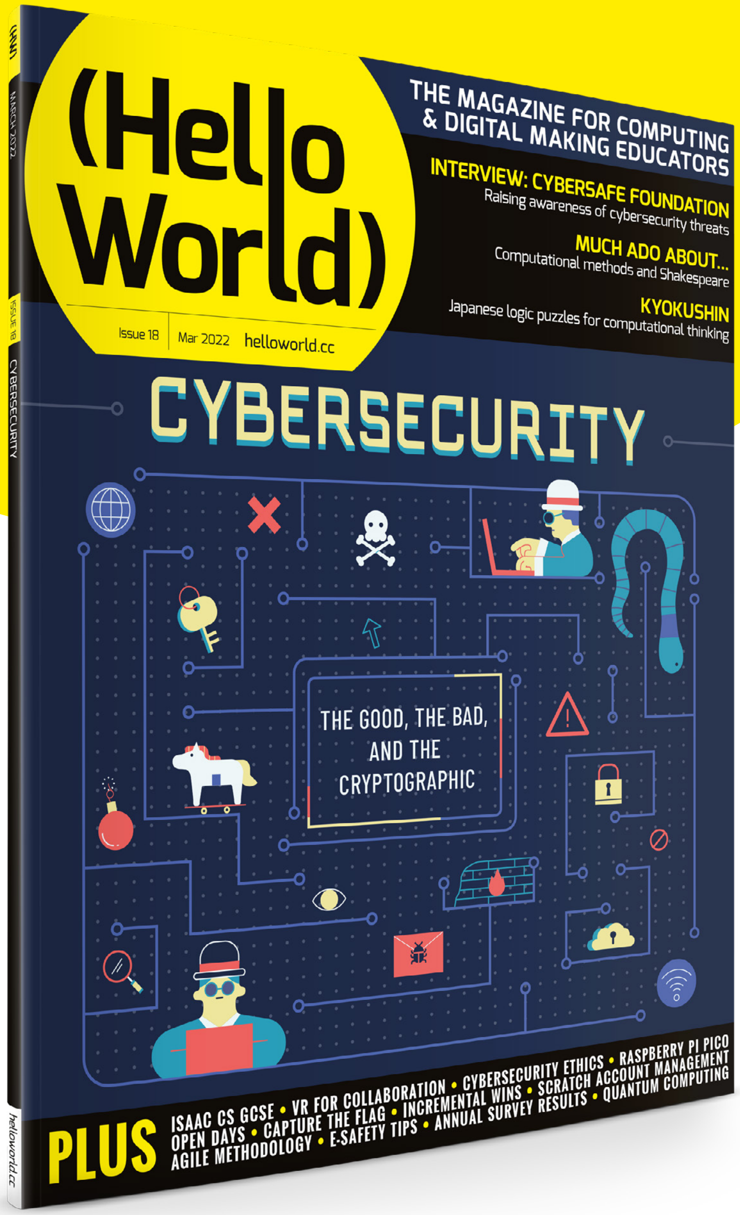
In the table below, you will find learning outcomes associated with the 'Design and development' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
Summarise the requirements, purpose, and audience of a task	Develop success criteria based on a project brief and user research	**Describe different approaches and methodologies to software development, such as prototyping and the agile approach**
Analyse existing solutions to identify what is possible and what 'good' looks like	**Research around a problem to determine the size, feasibility, and scope of a planned solution**	Identify a problem and produce a clear brief that includes measurable objectives
Identify appropriate planning techniques for a given project	Decompose a problem into manageable chunks and produce project plans	**Research a problem, making recommendations as to its feasibility, benefits, and risks**
Develop and communicate plans for projects using formal structured formats	**Build a project, recording versions and documenting decisions**	Produce comprehensive plans, detailing all aspects of a project
Build limited prototypes that prioritise the required features of a solution	Develop and collaborate on a project following a modular or staged approach	Generate and apply test plans to digital and physical artefacts
Choose design assets to suit the purpose and audience of a task	**Distinguish between different types of testing and their purposes**	**Explain the features that make a digital artefact maintainable and extendable**
Test finished artefacts alongside alternative solutions as part of evaluation	Select and apply a range of suitable tests to a digital artefact	Explain what is meant by the functionality, effectiveness, usability, and reliability of a digital artefact
Compare an artefact with the task and user requirements	**Evaluate a project using previously defined success criteria**	**Evaluate a project against its project brief and objectives**
Seek feedback from a broad range of individuals to improve a digital artefact	Gather feedback on the accessibility of a digital product in order to improve the user experience for all	Use collaboration tools to design and develop a digital artefact as part of a team

(HW)

SUBSCRIBE TODAY



(Hello World)
Issue 18
Mar 2022
helloworld.cc
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
INTERVIEW: CYBERSAFE FOUNDATION
Raising awareness of cybersecurity threats
MUCH ADO ABOUT...
Computational methods and Shakespeare
KYOKUSHIN
Japanese logic puzzles for computational thinking
CYBERSECURITY
THE GOOD, THE BAD, AND THE CRYPTOGRAPHIC
PLUS ISAAC CS GCSE • VR FOR COLLABORATION • CYBERSECURITY ETHICS • RASPBERRY PI PICO
OPEN DAYS • CAPTURE THE FLAG • INCREMENTAL WINS • SCRATCH ACCOUNT MANAGEMENT
AGILE METHODOLOGY • E-SAFETY TIPS • ANNUAL SURVEY RESULTS • QUANTUM COMPUTING

FREE IN PRINT for UK-based educators

Why subscribe?

- Teaching resources and ideas used by over 90 percent of our readers
- Exclusive news, research findings, and in-depth features
- Delivered to your door four times a year

CONVERSATION PERSONAL ACCOUNT
YOUNG CODERS COMPETITION
Shaun McMahon describes his experiences of the coding competition that his school has won for the last three years
SCHOOL CLOSURES HIT ENTRY NUMBERS BADLY, BUT THE TEAMS WHO ENTERED SHOWED GREAT RESILIENCE

TO SUBSCRIBE VISIT:

helloworld.cc/subscribe

(Hello World)
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
INTERVIEW: CYBERSAFE FOUNDATION
Raising awareness of cybersecurity threats
MUCH ADO ABOUT...
Computational methods and Shakespeare
KYOKUSHIN
Japanese logic puzzles for computational thinking
Issue 18 Mar 2022 helloworld.cc
CYBERSECURITY
THE GOOD, THE BAD, AND THE CRYPTOGRAPHIC
PLUS ISAAC CS GCSE • VR FOR COLLABORATION • CYBERSECURITY ETHICS • OPEN DAYS • CAPTURE THE FLAG • INCREMENTAL WINS • SCRATCH ACC AGILE METHODOLOGY • E-SAFETY TIPS • ANNUAL SURVEY RESULTS • QU

Not a UK-based educator?

- Buy a print copy from **helloworld.cc/store** — we ship to over 50 countries
- Subscribe to receive the free PDF on the day it is released
- Read features and news at **helloworld.cc**

FREE PDF

for anyone, anywhere

TEAM DIVERSITY AS A PREDICTOR OF INNOVATION

STORY BY Thom Kunkeler

Team diversity is one of the core advantages of learning in groups, and new research has shown its benefits for project innovation. The study, conducted at the Israel Institute of Technology, found that the inclusion in classroom settings of people from various academic disciplines and levels helps students create innovative and implementable solutions. Individual differences, researchers found, invite students to approach situations in various ways, and stimulate new ideas and fresh perspectives.

Over the past few decades, student populations have become increasingly mixed on biodemographic variables such as gender, age, and ethnicity. Although barriers to participation in education still persist, classrooms are increasingly mixed, especially in online settings.

For this study (**helloworld.cc/usher2019**), Usher and Barak were interested in how diversity relates to project innovation, both in face-to-face university settings and in online classrooms, for the course 'Nanotechnology and nanosensors'. The diversity of a team was rated according to four variables: gender, native language, academic discipline, and academic level. Not surprisingly, the group of online learners was more diverse than the group of university students, with followers from over 150 countries. The group of university students, although less diverse in native language than the online group, showed slightly more diversity in gender.

To understand how diversity relates to project innovation, both groups of learners were split up into teams of four and were asked to develop a new product. In the creative stage, students had to create new ideas, and in the application stage, the implementation was carried out. The innovation of students' team projects was assessed on product necessity, STEM interdisciplinarity, market readiness, and innovation type.

The results of the study indicate that diverse teams were rated more highly on project innovation, both for online and face-to-face learners. This evidence supports the idea that working in collaboration with people from diverse backgrounds enhances creative ideas and innovative solutions. In particular, the study found that a mixture of academic discipline and academic level — the task-related diversity — was key to project innovation. For teachers, this type of research should be taken as an incentive to experiment with classroom diversity for group projects. So, how diverse is your classroom? (HW)

FURTHER READING

- ✓ **Usher, M., & Barak, M. (2019). Team diversity as a predictor of innovation in team projects of face-to-face and online learners. *Computers & Education, 144*. helloworld.cc/usher2019**

Working in collaboration with people from diverse backgrounds enhances creativity

ONE DESIGN, THREE WAYS

Matthew Wimpenny-Smith explores the importance of implementing design in programming lessons through a project with his upper-primary learners

Design is very important, but can often be overlooked in computing projects, as we tend to be too quick to jump straight into the coding or making stage. Without good design and an understanding of how to implement it, though, we don't get good products or services — or code, for that matter! So why do we encourage children to be creators rather than consumers, without any consideration given to design?

Back in April 2018, I (Matthew) attended a fantastic three-day course run by Jane Waite called *Diving Deep into Primary Programming*, where there was much discussion and debate about how to implement design in computing lessons. For the 2019 academic year, I then set myself the goal of getting my pupils to really think about design and how to implement it.

Of course, spending time thinking about design required a shift in both my own mindset and that of the learners. This included helping them not to be tempted to just jump in and start coding, but to think critically and apply their computational thinking skills. They also needed to learn their own limitations, such as what is possible within the software and hardware they're using, how long they have to complete a project, and what skills they need to meet their design ideas — in other words, what is doable.

I'm very fortunate to teach computing across all year groups in my school. In the preceding six years we had done some design, but usually not with a clear purpose of what I was expecting, or how we would use the designs within the coding process. I therefore wanted to formalise this, as I'd noticed that spending time on design helps with the process of decomposition and the abstraction skills of computational thinking. Alongside this, we need to help pupils to evaluate software and hardware, and decide what might be the best way to implement a design.

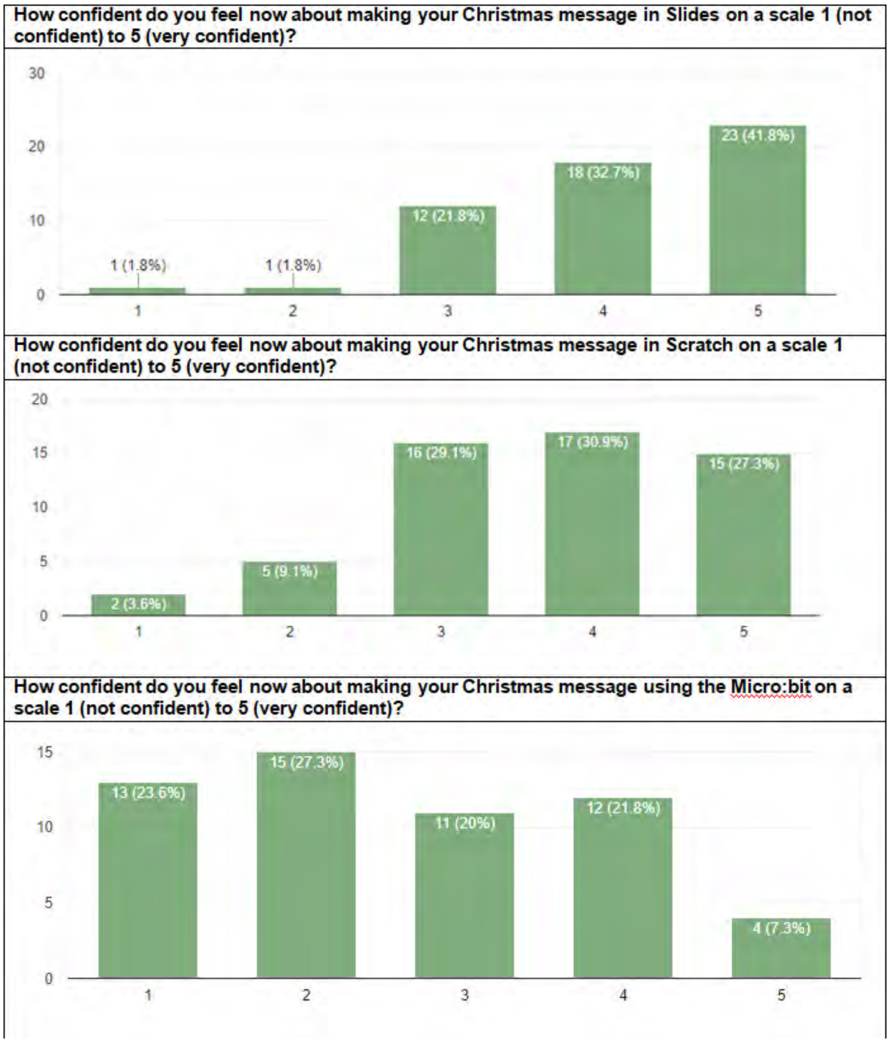


Figure 1 The results of the pre-assessment show levels of pupil confidence across the three systems

All images courtesy of Matthew Wimpenny-Smith

Christmas message project

It was a sunny October day in London and, as I was having lunch with Jane, the conversation turned to what I was planning to teach my Year 6 pupils (aged ten to eleven) after half term. I said, "I am thinking about starting to transition them from blocks to text coding." Jane asked, "Are they ready to move on?" and warned me not to do it too soon, emphasising the need for a firm foundation of block-based programming. I was intrigued to find out whether they were ready, so over that lunch meeting, we thrashed out the idea for a design implementation project that would give me more information on what they currently knew and could do.

The project asked a simple question: could the pupils create a design and independently implement it in a variety of software that I felt they were proficient in, including block-based code, and could I then move them on to a text-based language using the same design?

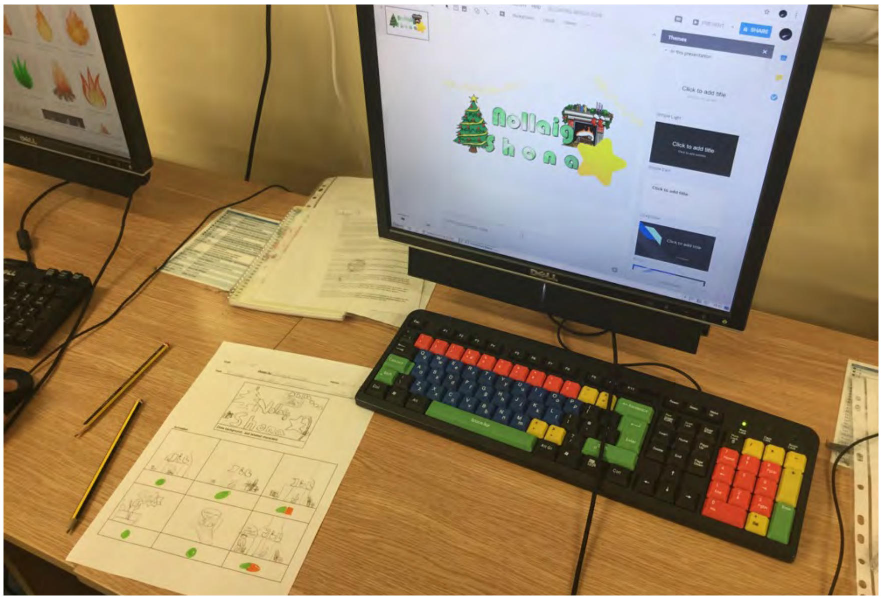
Christmas was approaching, so we settled on the idea of creating a Christmas message for other pupils in the school, with the added language twist of it being in different spoken languages, such as French, Japanese, and so on.

I chose Google Slides, Scratch, and a micro:bit for the implementation, then started with a pre-assessment to find out levels of pupil confidence within the three implementations. The results of this assessment showed without a doubt that they felt most confident using Google Slides, followed by Scratch, and then the micro:bit, as highlighted by the bar graphs in **Figure 1**.

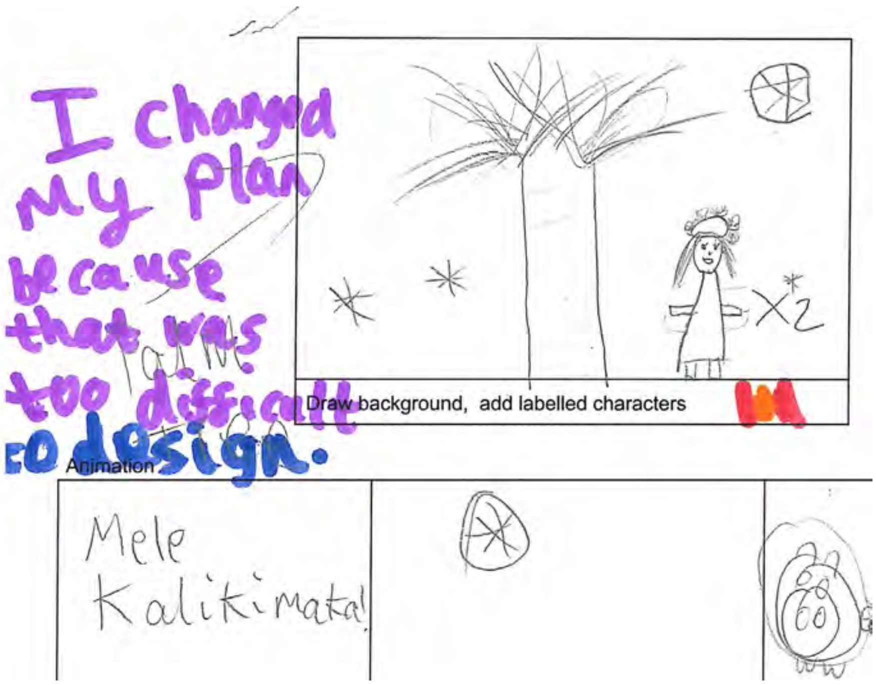
The results of this are perhaps predictable, mainly because of pupils having lots of exposure to Google Slides across all their lessons, and limited exposure to Scratch, mainly via my computing lessons (one hour per week, plus one hour of Code Club for those who were keen). And only those who attended my Code Club or had enthusiastic parents or siblings in the senior school had any exposure to the micro:bit before this project. I did, however, briefly demonstrate the micro:bit just before the pre-assessment.

Design stage

After the pre-assessment, the pupils worked on their paper designs, followed by a self-assessment for confidence using red, amber, green (RAG). I set them a challenge to implement their design as closely as possible within the three different systems (Google Slides, Scratch, and micro:bit). Between each implementation, I photocopied their designs and asked them to reassess them for confidence, and also to make notes on how they would need to change their designs to fit the limitations of the software. This helped to highlight their initially ambitious ideas and hone their critical thinking around what was actually possible, given their knowledge and understanding plus the curriculum time constraints.



The pupil design being implemented in Google Slides


I changed my plan because that was too difficult to design.
Draw background, add labelled characters
Animation
Mele Kalikimaka!
x2

The pupil design being reviewed based on the implementation

Implementation stage

The pupils found they could implement their designs with little change in both Slides and Scratch, drawing on their existing knowledge (you can watch a video of the implementation in Scratch at **helloworld.cc/scratchimplementation**). However, I realised after pre-teaching how to use the micro:bit that this was a different approach, moving away from a block-based language to a text-based

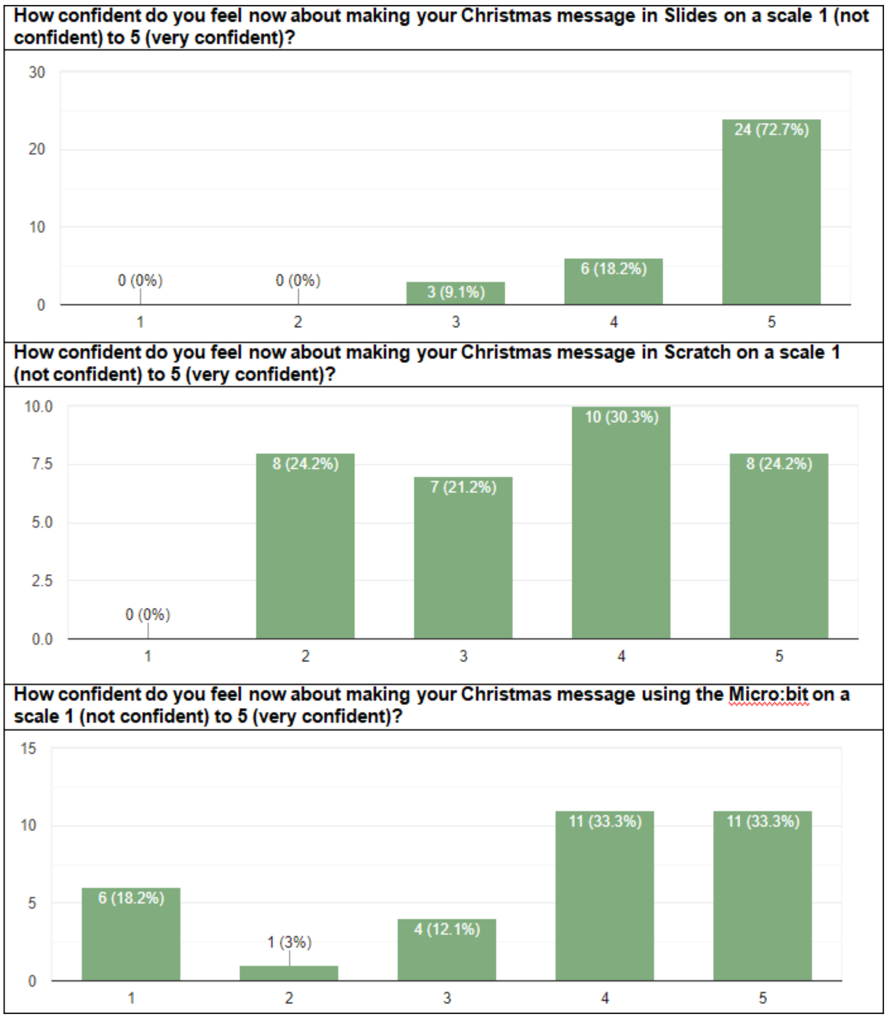


Figure 2 Summary of the post-assessment survey results

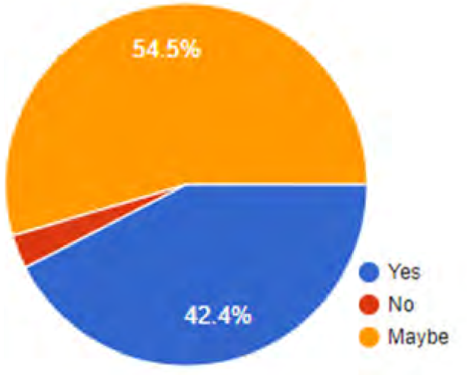


Figure 3 Pie chart showing how many of the pupils felt they were ready to move on from Scratch

MATTHEW WIMPENNY-SMITH

Matthew is leader of digital strategy and a computing subject leader. He has worked for Headington School Oxford, UK, for the last seven years in the Prep School, teaching EYFS, Key Stage 1, and Key Stage 2. He is a CAS Master Teacher and the Oxfordshire Primary Community Leader. He is also a BCS Certified computer science teacher, Raspberry Pi Certified Educator, Google L1, and NCCE facilitator (**@MWimpennyS**).

JANE WAITE

Jane is a research scientist at the Raspberry Pi Foundation. Her interests include using design in primary programming, semantic waves, PRIMM, and migrating to online teaching using ABC (**@janewaite**).

interface, and that their designs would need to be modified. I purposefully kept them constrained to just being able to display and scroll through text and images on the micro:bit LED matrix, and I would use the text-based Mu editor. The next three lessons were a busy time for the class as they adapted their designs and taught themselves how to scroll and display images on the micro:bit. They also discovered that there were limitations, such as not being able to display certain languages, for example those with Cyrillic-based alphabets.

Post-assessment

The post-assessment survey highlighted some interesting results (**Figure 2**). Predictably, the pupils' confidence with using Google Slides was high, but interestingly, there was a shift in how they felt about Scratch. It must be noted that for this project, I intentionally gave limited support with both Slides and Scratch.

The results and comments from the post-assessment suggested that pupils found following a design constraint in Scratch harder than they initially thought. This led me to realise that they overestimated their abilities with Scratch. One pupil who found it challenging even asked, "Please can I just do the whole project in Slides?"

Perhaps pupils may have been overconfident with Scratch, as in the past, they've tended not to stick to an initial idea once they got started, and just done what was easy as they wrote the code; or perhaps they are used to copying code without truly understanding it. Scratch is complex, and learning how to implement a design isn't easy. I need to work more on this, to show them how to implement ideas and know what is doable. It seems that the current way I teach Scratch isn't quite hitting the mark, and I need to do more research in my class to explore this.

This was emphasised by the results of another question, this time asking them if they felt ready to move on from Scratch (**Figure 3**). The results show that over 50 percent were either not sure or not ready. This classroom research has highlighted the need for me to ask my pupils about their learning more often, and not to move them on until they're ready. I also need to do more to develop my pedagogy of teaching computing — but that's for another article! (HW)

AGILE METHODOLOGIES IN THE CLASSROOM

Introducing agile practices into the classroom has lots of benefits beyond giving students a taste of what working as a software developer entails

Being agile means moving and responding quickly, and it's also an adjective that is applied to a range of methodologies developed to improve the software development process and life cycle. It means being able to respond to shifting requirements, industry changes, and unforeseen problems. It usually involves cross-disciplinary teams of developers collaborating, planning, and reviewing software together and one of its key tenets is to work in short iterations, building the minimal product you need and then refactoring and improving it later.

It stands in contrast to the more traditional 'waterfall' approach, in which a product flows down through several fixed stages, from initially deciding the requirements, through design, building, testing, and then finally delivery. Years might have passed between the first and final stages. Years is a long time in the tech world, and by that time, the requirements might well have changed. But too late — the product has already been signed, sealed, and delivered.

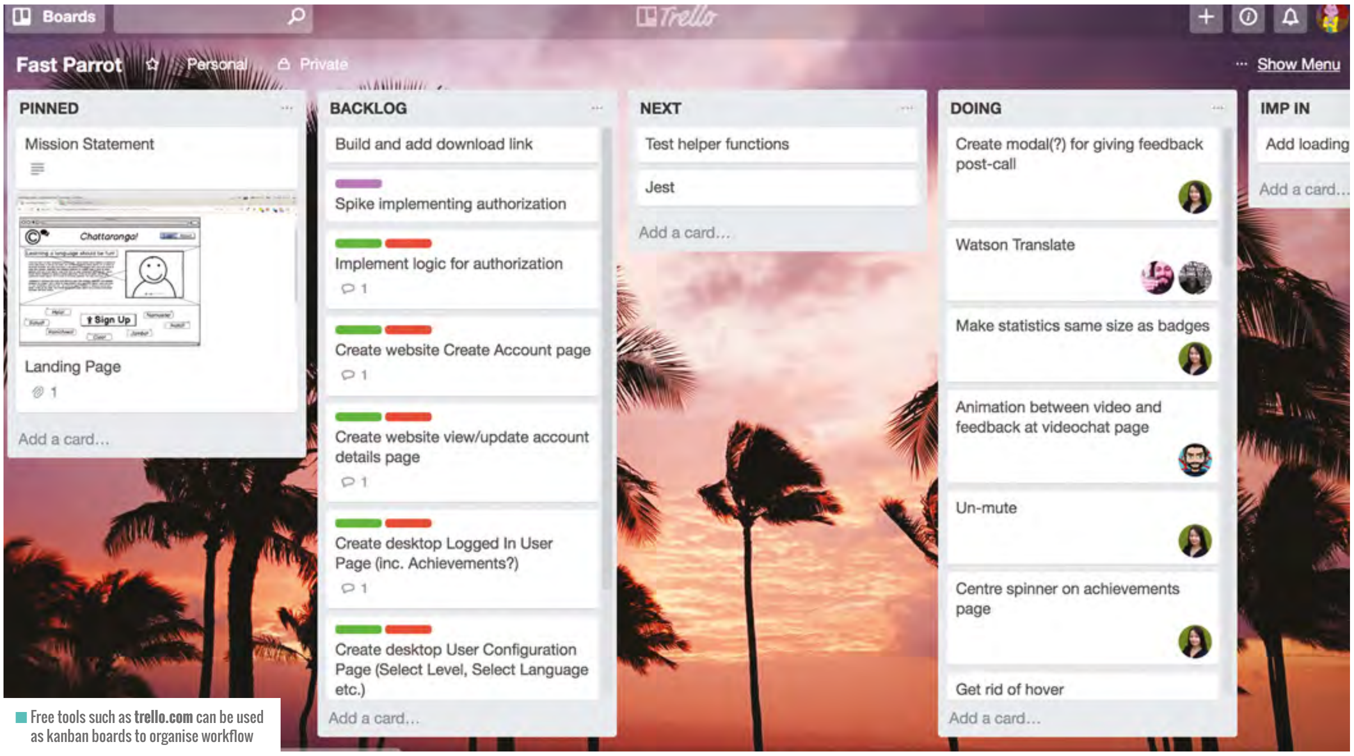
Why in the classroom?

I teach software development to adults, preparing them for work in the industry. Exposing them to common industry practices and structuring their classroom like a development shop is important in preparing them for work. However, the techniques and ideas that I'm going to talk about can be applied to all areas of life, and provide tools for tackling any project or problem.

There is a whole range of methodologies and frameworks that can be used to help you work in a more agile way. I won't go into all of them — just a few favourites that we employ in the classroom every day.

Pair programming

This is the most important agile methodology we use. During a project (which we call a 'sprint' and lasts two days), we pair students for about half the time. We initially give students a quarter of the allotted coding time to explore aspects of the problem alone, and then we randomly pair them up to continue working. When they pair, they have to start again, integrating ideas from both of their initial explorations. At the end, they split again and independently finish up the project in whatever way they prefer.



PINNED
- Mission Statement
- Landing Page
- Add a card...

BACKLOG
- Build and add download link
- Spike implementing authorization
- Implement logic for authorization
- Create website Create Account page
- Create website view/update account details page
- Create desktop Logged In User Page (inc. Achievements?)
- Create desktop User Configuration Page (Select Level, Select Language etc.)
- Add a card...

NEXT
- Test helper functions
- Jest
- Add a card...

DOING
- Create modal(?) for giving feedback post-call
- Watson Translate
- Make statistics same size as badges
- Animation between video and feedback at videochat page
- Un-mute
- Centre spinner on achievements page
- Get rid of hover
- Add a card...

IMP IN
- Add loading
- Add a card...

Free tools such as trello.com can be used as kanban boards to organise workflow

Students driving and navigating on a project

DRIVING AND NAVIGATING

The driver:
- Is the one using the computer
- Listens to the navigator
- Implements the ideas of the navigator
- Can question the navigator

The navigator:
- Directs the driver
- Explains their ideas
- Doesn't touch the computer
- Can use pen and paper to help explain their thinking

How to pair

While pairing, students work together on one of their computers and follow a strict driving–navigating model, as explained in the boxout. With this set-up, students have no choice but to try to explain and communicate their ideas. This can be challenging at first, but continual practice at explaining technical concepts and ideas helps students embed what they're learning and discover the areas they don't fully understand.

> PRACTICE AT EXPLAINING TECHNICAL CONCEPTS AND IDEAS HELPS STUDENTS EMBED WHAT THEY'RE LEARNING

Every 30 minutes, we ask students to swap roles. This might involve using a version-control system to push their work to a service such as GitHub and then pull it down on the other student's computer, or it could just mean swapping who is using the same computer.

Scrum

Scrum is a framework for managing roles and workflow in an agile way. A few things we've borrowed from Scrum are the concepts of stand-up meetings and retrospectives. If students are working on a project in a small group over several days, we help them organise stand-up meetings at the beginning of the class, where they stand in a circle and each student has a few minutes to explain what they've been working on, what they found challenging and whether they've overcome it, and what they want to achieve today. We use this in combination with kanban (explained below). We think it's important for every student to hear what other students struggled with, as everyone will be struggling with something.

We also encourage group retrospectives, where upon finishing a sprint, the group will discuss what went well, what didn't go well, and what could have been done differently. Sometimes, students will keep ongoing blogs or logs of their work on a project and use this to refer back to during retrospectives.

Kanban

The principle of kanban methodology is to be able to visualise the work and tasks to be completed. Kanban means 'billboard' in Japanese and that's usually what you use — some kind of billboard divided into columns. Tasks can be moved from column to column as they progress through the development workflow. Sometimes, the board is drawn up on a whiteboard, but free software such as Trello (**trello.com**) can also be used. You can tag individuals on cards, add additional information, and upload attachments.

GitHub has a similar board system called Projects. GitHub Projects are great because you can integrate your cards with GitHub Issues and link to parts of your code.

Many students on our course end up using kanban to organise everything, from studying, to job hunting, to general life!

'Just make it work'

Another concept we spend a lot of time talking about is the minimum viable product (MVP). Building an MVP means building the most bare-bones version of the product that will work. We discourage students from trying to create the most amazing, all-bells-and-whistles product on the first attempt, because who knows how the specification might end up changing later and how much time we will have wasted.

Beyond practical implication, stressing the importance of 'just make it work' encourages small wins, which boost confidence. Students can improve, refactor, and rethink their initial attempt, but at least they've got something to show for their work so far. Creating something that 'just works' also gives us a starting point to talk about how we can make the code cleaner and more modular, or to consider edge cases. (HW)

HARRIET RYDER
Harriet teaches software development to adults in one of the UK's leading boot camps.

TESTING: THE ~~FEAR~~ LOVE OF FAILURE

Pete Bell offers tips for inculcating a culture of testing (and failing!) in the computing classroom

One of my fondest memories of teaching is when a group of ten-year-old students visited my secondary school for a computing experience day. The first question I asked students was, "What does FAIL stand for?" After I'd given them the answer (First Attempt In Learning), I spent a few minutes telling them that the second time they fail, it changes its meaning to "Further Attempts in Learning" and that, each time they fail, it just means they are learning how to improve. Throughout the day, I rewarded students who told me they'd got something wrong. At the end of their experience day, I walked them to the minibus, and the head teacher came over and asked them what they'd been doing. One young lad shouted passionately "We've been learning to FAIL!" I'll admit to being a bit nervous about what the head's reaction might be, but he just grinned and said "Brilliant!"

In our culture, we reward people for doing things right and punish them for getting them wrong. Wrong is bad; right is good. Do you recognise this in your students? 'I tested my product thoroughly and found there was nothing I could improve.' This might not be a result of laziness or a lack of understanding; it could just be a student wanting to show that they got it all right, because 'right' is 'good', so that means they did 'good' — right? So, because testing is (mostly) the process of identifying things that are wrong, the challenge is: how do you help students unlearn that wrong is bad?

Normalising testing

It's important to instil a love of failure and improvement in your classroom, and we can do this by normalising testing. Here are a few tips:

Don't test only code: it seems simple to build testing into upper-primary or lower-secondary programming sessions. However, when programming is still so new as a concept, it is tough for students to think about what might be wrong with their work: they don't know what they don't know.

Instead, you could initially plan a lesson on testing something that's more familiar to students, like some pre-prepared directions to the school library for an open day, or for someone with a visual impairment. If you add in a few mistakes along the way, it will give students a chance to find errors and correct them. You could also test a piece of writing, a drawing, a piece of music, or a mathematical equation.

Model how to fail well: in my first interview for an ICT teacher role, I couldn't turn the projector on. Embarrassing? Yes! However, in my feedback, I was told that part of the reason I was appointed was the way in which I told the students that I'd made an error and then calmly talked them through my process of checking (testing) each potential cause as I went about solving the problem.

Make it explicit early: build debugging into lessons all the way from primary and lower secondary, so that it becomes a standard part of the programming process.

Build a culture of support and improvement: programming lessons should build confidence. One method you can use is pair programming, which has been shown to improve social and communication skills (see the article on page 58 of *The Big Book of Computing Pedagogy* for more on this). However, allow a little longer for students to find a solution through collaboration.

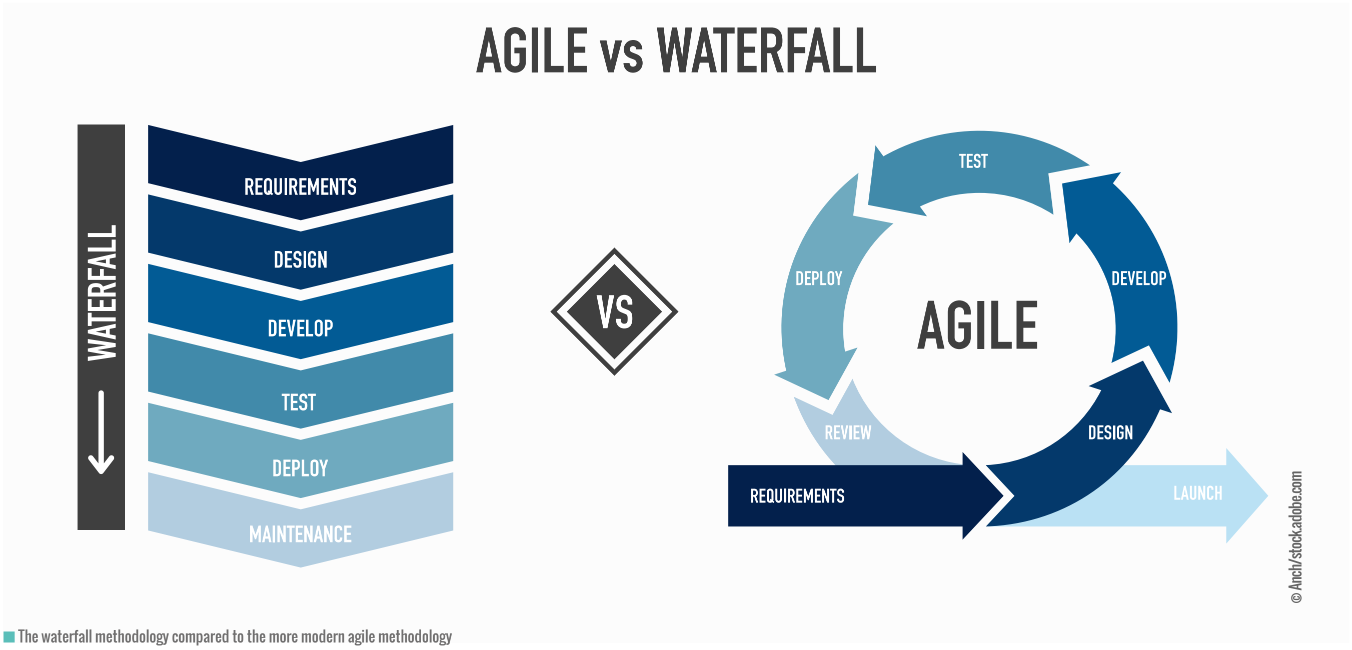
Model it and scaffold it: provide students with a test plan for a short piece of code and demonstrate how to use it. Have a look at the test plans section of Isaac Computer Science for examples (**helloworld.cc/testplans**).

The testing process

Coursework projects and non-examined assessments have historically been structured to reflect the waterfall development methodology, where testing is a process that's carried out after development. More recently, mark schemes for these types of project have been updated to reflect more modern agile methodologies, in which students are expected to apply agile development principles, with testing assessed throughout the process (see the previous article for more on agile methodologies).

> INSTILLING A LOVE OF FAILURE AND IMPROVEMENT IS IMPORTANT, AND WE CAN DO THIS BY NORMALISING TESTING

In a programming lesson, modelling failure could take the form of a live-coding demonstration. Almost every time I have shown students a programming concept during one of these demonstrations, the thing doesn't work as I thought it should … and that's more than OK!



■ The waterfall methodology compared to the more modern agile methodology

So how can students best show their understanding of testing for formal assessment? Let's walk through it by using an advanced-level project (for students aged 16+) as an example, where students submit a coding solution to an identified problem, along with a report explaining their testing process:

- In their project report, students should demonstrate how they have put theory into practice. From the start, students should be able to explain their approach to testing — justifying an agile approach, for example.

- Having analysed and decomposed the problem, students could then identify a range of (not necessarily all) data that could be used in acceptance testing to show that the system works as intended, including validation tests to check the robustness of some of the planned subprograms. Students can show how they have embedded testing holistically by testing a few of their designs with end users and updating those designs based on feedback.

- Students should be selective when choosing subprograms to test, focusing on those that can provide opportunities to best demonstrate good testing practice. Examples are login systems, timers, scoring systems, storage and network requirements, or specific uses of a particular data structure. Some subprograms or classes will be very similar, so it's best to test more than one of the same type only if it allows a student to show a new testing concept.

- In addition to the more obvious opportunities to carry out iterative unit tests, it is important to document integration testing to check how a few modules work together, preferably ones with interesting edge cases. You can read more about types of testing on Isaac Computer Science at **helloworld.cc/testingtypes**.

- When a solution has been through this alpha phase, end users can then carry out some usability testing — approaching this as a beta-test opportunity that is again cross-referenced to the success criteria identified at the analysis stage.

An advanced-level student wants to see where they went wrong and how they fixed it. Although it's not always part of the assessment criteria, it is important for students to reflect on their journey as learners, including how they have developed wider twenty-first-century skills, such as critical thinking as they planned and executed their tests; creativity as they developed their solution; collaboration as they supported their peers during code reviews; and communication as they articulated their coding journey in a report. Above all, they should have built resilience as they planned and iterated their development as a result of testing feedback.

In the end, isn't education all about learning that failure is OK? With that in mind, there was no need for me to be so nervous about my head teacher's reaction to my group's exclamation that they'd been failing with me all day. It's his job to see where improvements can be found; he was just testing. (HW)

PETE BELL
Pete is a learning manager at the Raspberry Pi Foundation. He was previously a computing teacher for 23 years and the head of his school's STEM faculty. Pete is also an experienced assessment designer (**@petejbell**).

SAFETY AND SECURITY

Like most tools, computing systems promise many advantages, benefits, and opportunities — but they are not without their risks. A vital part of computing education is for learners to develop their understanding of these risks and the steps they can take to mitigate them. This topic area has a few different dimensions to it that are more or less relevant at each learning stage.

Early on, we need to support learners in becoming safe users of digital technology, so that they can realise the benefits of such technologies without putting themselves at risk. While online personal safety should be present in every school subject area, computing classes provide a space for learners to understand the limits of computing and how technology can be exploited.

Personal safety lessons continue to thread through all learning stages, with the focus shifting to the risks and threats relevant to learners' ages and the technologies they use. However, as learners progress, they should also begin to consider safety and security from an organisational and societal standpoint, as well as delving into the technical innovations that help keep systems secure.

IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** safety and security, in summary
- **What the research says:** media use and attitudes
- Online safety teaching
- Safeguarding in online lessons
- Cybersecurity activities and tools
- Thinking like a hacker

IN SUMMARY

SAFETY AND SECURITY

Understand the risks of using technology, and how to protect both individuals and systems

STAGE 1

- Explain how to use computing equipment safely
- **Use information technology safely and with respect**
- Demonstrate safe behaviour when working online
- **Give examples of personal information and know who it should and should not be shared with**
- Outline how individuals should report concerns about content or contact from the internet

STAGE 2

- Demonstrate safe and responsible behaviour when using a range of information technology
- **Demonstrate how to keep passwords and other credentials safe and secure**
- Describe when individuals might be asked for personal data online
- **Explain the security risks of linking to or sharing content owned by others**
- Explain how misleading information or media can pose a security risk

In the table below, you will find learning outcomes associated with the ‘Safety and security’ strand of the Raspberry Pi Foundation’s computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3

- Explain the attributes and practices that make a password secure
- **Explain how online activity leaves a lasting digital footprint**
- Explain common techniques used to exploit individuals and steal personal data
- **Describe how users can control where and with whom their personal data is shared**
- Describe common threats to computing systems and ways to protect against them
- **Understand the risks and related precautions associated with online services**
- Explain the purpose of encryption along with examples of its use
- **Identify where and how to report inappropriate content, contact, or conduct**

STAGE 4

- Explain how safety and security principles apply in emerging contexts or technologies
- **Describe examples of some vulnerabilities of computing systems, networks, and software**
- Recommend security measures and practices for different scenarios
- **Explain the need to balance security with other considerations including usability, cost, and ethics**
- Describe a range of malware and the threats that it can pose
- **Explain the rights and responsibilities that individuals and organisations have with regard to personal data**
- Outline the role of encryption in protecting data and its importance in data security
- **Describe authentication techniques used to improve security, such as biometrics, captchas, and two-factor authentication**
- Describe the strategies that organisations can use to keep their systems and information secure

STAGE 5

- Explain the concept of computational security
- **Explain what a vulnerability is and how it is introduced or exploited in a system**
- Explain how senders and messages can be authenticated using digital signatures and certificates
- **Describe the process, applications, and limits of modern cryptography techniques**
- Apply and compare a range of ciphers to encrypt and decrypt data
- **Attempt to decipher encrypted data applying common cryptanalysis techniques**
- Distinguish between symmetric and asymmetric encryption schemes
- **Describe strategies and techniques to protect against common security vulnerabilities**

ONLINE SAFETY: WHAT DO LEARNERS KNOW?

There's a large gap between what learners think they know about online safety and what they actually know, but some small changes to classroom focus can make all the difference

STORY BY Sway Grantham

Teaching online safety is hard. Curriculum requirements vary from class to class, from year to year, and even from one term to the next. Teachers have to work hard to ensure they know how to use specific tools safely, how to support young people, and that the tools they're using are still relevant. On top of this, there are the broader trends in terms of what tools students are using. Each year in the UK, Ofcom (the UK government's communications regulator) produces a report about children and parents' media use and attitudes, which highlights some of these trends. In the 2022 report, 99 percent of 3–17-year-olds had been online in the last year (helloworld.cc/ofcom2022). While this fact is perhaps no surprise, pair it with research conducted by Peter Macaulay and colleagues concluding that students think they are better at keeping themselves safe online than they really are (helloworld.cc/macaulay2020), and it appears we need to review our online safety curriculum focus.

How are learners using tech?

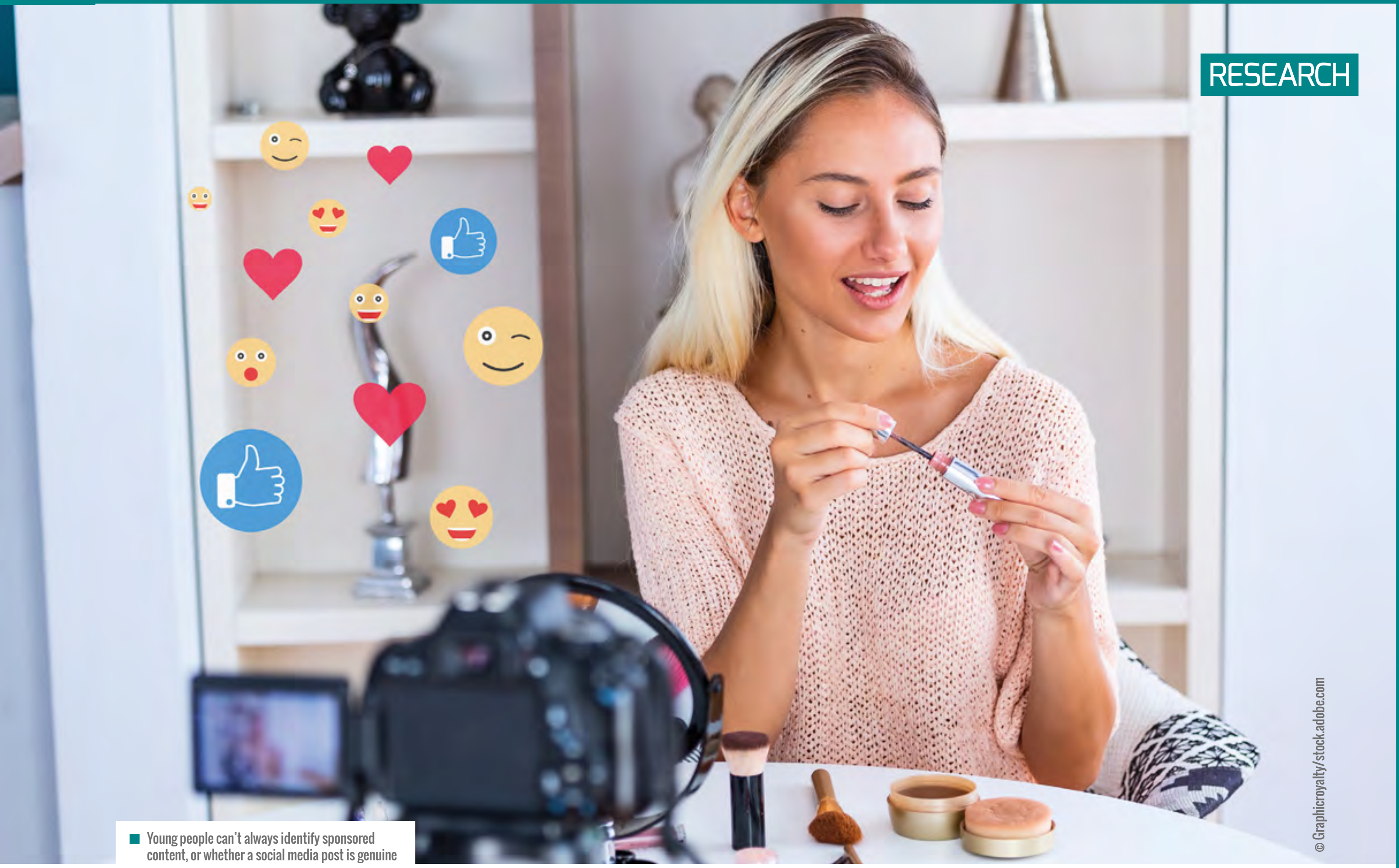
A helpful first step in providing suitable online safety education is understanding how technology is being used by learners of different ages. Classes will vary, but the Ofcom report suggests that children aged 3–4 largely use video-sharing platforms (89 percent) and watch TV and films online (81 percent). These continue to be popular activities for those aged 5–7 (95 percent and 74 percent respectively). By the 8–11 age group, 84 percent are using messaging sites and apps as one of their main online activities. This trend continues with those aged 12–15, with 91 percent using social media and 73 percent using live-streaming apps and websites. Finally, by 16–17, 100 percent had a mobile phone and this was their primary device for interacting online.

This high level of exposure to the internet can lull educators and parents into a false sense of security; they may think that young people therefore have the skills they need to navigate this world successfully. And to an extent, they are right. These tools are designed to be intuitive, and through experience, people learn how to use them more proficiently. However, we don't want learning about online safety to happen through experience. Knowing there is a report button on a website does not mean you know what you should report, what behaviour is unacceptable or inappropriate, or why you should report it. This conceptual understanding needs to underpin online safety education.

What do learners actually know?

Researcher Macaulay and his colleagues used questionnaires with learners aged 9–11 to compare their subjective knowledge of online safety to their objective knowledge. Subjective knowledge is how learners feel about their own ability, while objective knowledge reflects their actual understanding.

To measure subjective knowledge, learners were asked to rate statements such as 'I know what to do to stay safe on the internet' and 'I know what things could put me in danger or upset me on the internet' on a four-point scale, from 'Disagree a lot' to 'Agree a lot'. To then measure their objective knowledge, learners were asked open questions such as 'What things might put someone in danger of harm, or make them feel upset, when they use the internet?' and 'What things can you do to stay safe from harm or getting upset on the internet?' The learners' ideas about the dangers of the internet were then grouped into six categories of the most common responses: people online pretending to be somebody else; being in contact with



Young people can't always identify sponsored content, or whether a social media post is genuine

people we do not know; sharing personal information/personal photographs or videos; cyberbullying; inappropriate and/or distressing content; and computer viruses.

This study found that children felt safe online, and that their subjective knowledge was high. This was especially true of boys and older children (in this study, 'older children' were eleven). In contrast, when their objective knowledge was tested, students could only identify, on average, two out of the six categories of online safety risks. When faced with these two conflicting sets of knowledge — what learners think they know, and what they actually know — children's subjective knowledge was the biggest predictor of how safe they felt they were online. This is worrying, as it may lead students to wrongly conclude that they are internet-savvy and get complacent when considering the risks they are taking online.

Improvements with age?

There is not yet an equivalent study for learners aged twelve and above, but the Ofcom report can give us some insights. As part of their report, Ofcom asked young people what features indicated that a social media post was genuine, whether they could spot a fake social media profile, and how they identified sponsored search results and content from influencers. Unfortunately, objective knowledge about online safety doesn't seem to improve massively with age, with the numbers varying very little between the 12–15 and 16–17 age groups.

Overall, these young people were best at recognising fake profiles (64 percent and 65 percent respectively). Few of them were able to identify sponsored content, either in searches or from influencers (38/39 percent for 12–15 year olds and 44/48 percent for 16–17 year olds), and very few were able to recognise whether a social media post was genuine (11 percent and 13 percent respectively). For age groups in which six out of ten children stated that they used social media to get their news, this is clearly a problem that needs to be addressed through education.

> A HIGH LEVEL OF INTERNET EXPOSURE CAN LULL TEACHERS AND PARENTS INTO A FALSE SENSE OF SECURITY

Changing online safety education

Many of the strategies outlined in the article on page 146 would help move online education towards where it needs to be. However, there are also some specific pedagogies and approaches that are particularly relevant to ensuring we are improving learners' objective knowledge and not just their subjective knowledge.

LEAD WITH CONCEPTS

Teachers can often fall into the online safety trap of focusing on teaching a process, such as how to report inappropriate comments. However, what is missing here are the concepts underpinning the process to ensure attitudinal change. Understanding what comments to report and why you should report comments is much more powerful than just learning about the actual process.

© AlesiaKan/stock.adobe.com

Even if a child isn't old enough to use a particular platform, it can still be beneficial to teach them about its dangers

MAKE CONCRETE

There is often a nervousness among teachers when it comes to teaching learners about being safe online, due to potentially encouraging uptake among those who do not already use the tools, or being aware that they do not meet the stipulated age to use the tool being discussed. However, we teach children to cross roads safely before we expect them to do it independently. The more skills we can give learners that are grounded in real-life, concrete knowledge rather than theoretical knowledge, the easier it is for them to apply them. This could be through using social stories and debates to discuss morals and ethics, or by using role play, for example becoming data detectives to find 'hidden' information about someone through the content they post online.

ASSESS ONLINE SAFETY

What this research highlights is that the confidence with which learners will discuss online safety is not a fair assessment of their actual understanding. By teaching key concepts and using concrete examples, there should be content that you can objectively assess to inform future approaches to teaching and learning. There are some examples in the research shared in this article, and others have been built into resources, such as in the Raspberry Pi Foundation's Teach Computing Curriculum units (for example, the unit of work for students aged ten to eleven at **helloworld.cc/TCCcommunication**).

"THE CONFIDENCE WITH WHICH LEARNERS DISCUSS ONLINE SAFETY IS NOT A FAIR ASSESSMENT OF THEIR UNDERSTANDING

These changes to educators' practice are not substantial. They do require some preparation and some time spent understanding the learners in your classrooms each year, but as teachers, you do that every day. However, actively making objective online safety a focus could prevent a student from learning about the risks of online safety the hard way. (HW)

FURTHER READING

- ✓ Macaulay, P. J. R., et al. (2020). Subjective versus objective knowledge of online safety/dangers as predictors of children's perceived online safety and attitudes towards e-safety education in the United Kingdom. *Journal of Children and Media, 14*(3), 376-395. **helloworld.cc/macaulay2020**
- ✓ Ofcom. (2022). Children and parents: media use and attitudes report 2022. **helloworld.cc/ofcom2022**

LESSONS FROM THE CYBERSECURITY INDUSTRY

STORY BY Katharine Childs

Cybersecurity topics in school computer science curricula often focus on defending against cybersecurity attacks and understanding the ethical and societal implications of data privacy. In programming topics, however, a proactive approach to writing secure code is also important. A research project called Motivating Jenny (motivatingjenny.org), supported by the UK's National Cyber Security Centre, has created a number of tools to help developers consider the security of their code and embed a workplace culture in which software security is seen as a fundamental value. Although this research has been conducted in industry, there are many ways in which educators can translate the findings into good classroom practice.

Using prompt cards and values cards can help to stimulate positive conversations about cybersecurity issues

© successphoto/stock.adobe.com

Real-world industry perspectives

In a 2019 study by the Motivating Jenny project, researchers took an ethnographic perspective. An ethnographic research study aims to find out more about the behaviours and routines of a group through direct observation, and researchers conducting this study met with professional developers in their workplaces to find out more about their beliefs, attitudes, and experiences with writing secure code. The write-up of the study provides a real-world example of a computing career and is useful for both teachers and pupils.

The researchers ran a series of workshops with groups of professional developers, with the aim of finding out more about how social interactions and a culture of software development contributed to supporting developers in writing secure code. In the first part of the workshop, participants read various examples of a security compromise that had taken place, each with a different focus or perspective. They then all read a first-person account of the impact of the security breach. Finally, the workshop concluded with a discussion of participants' own experiences. Prompt cards and values cards kept the group conversations focused on attitudes, beliefs, and behaviours, to create discussions that were "refreshingly different" and "participatory".

Applying findings to the classroom

The findings from the workshops were designed to be applied to professional settings. However, the themes that emerged could be equally relevant to the computing classroom:

Personal stories resonate: cybersecurity can often seem like an abstract or intimidating topic. However, all software and systems are designed and implemented by people. Consider how to use examples of real-life security compromises, such as the first-person account that was used in the research, in teaching and learning (**helloworld.cc/firstperson**).

Values can support positive discussions: in the research, using values helped to create positive, non-confrontational conversations. Your school community is likely to have a set of values to help each pupil shape their understanding of the world. For example, a classroom wall display that links cybersecurity considerations to the school's values is a powerful way of modelling this and providing useful discussion prompts.

Group discussion through play is effective: the researchers gamified group discussions by using a timer and prompt cards. This provided a valuable structure to conversations and helped to give each participant a chance to speak. Consider how this approach could work when you are leading classroom discussions with your pupils. (HW)

FURTHER READING

✓ Lopez, T., Sharp, H., Tun, T., Bandara, A., Levine, M., & Nuseibeh, B. (2019, May 28). *Talking about Security with Professional Developers.* 7th International Workshop Series on Conducting Empirical Studies in Industry, Montreal, Canada. **helloworld.cc/lopez2019**

WHY DISCRETE ONLINE SAFETY TEACHING IS NOT ENOUGH

Sway Grantham shares suggestions on how to ensure your primary online safety curriculum is fit for purpose

Enter any primary school and you will find strategies for keeping yourself safe being modelled every day of the year. From not running with scissors to negotiating whether it's acceptable to declare who someone else can or can't play with at lunchtime, we are constantly guiding learners to recognise boundaries, be honest about their choices, and develop a little common sense that jumping off something really high may hurt. Yet, more often than not, I see online safety relegated to computing lessons and Safer Internet Days. One-off lessons are simply not enough to safeguard our learners.

We all want to keep our learners safe, and it's challenging to ensure our online safety curriculum stays up to date. I recommend asking yourself the following three questions when reviewing online safety in your school: "What's your context?", "What's trending?", and "Where is it relevant?"

What's your context?

It can be very tempting to start a school year or unit of work with several discrete online safety lessons. These often demonstrate to outside observers that online safety is taught in your school (there's a timetabled lesson for it!) and reassure you that learners have some safety skills before you begin using technology more prevalently. However, what we often see here is a lack of context.

For example, if we ask learners what personal information is, they can tell us it's their name, school, address, and phone number, and they know that they shouldn't share those things online. Then they share a photo of themselves on social media in their school uniform, in front of their house, with their best friend. Without the context, such as different ways and places we share information about ourselves online, learners often imagine that the only time they need to be careful about this is when someone is directly asking for their contact information.

What's trending?

The digital world that your learners are a part of is constantly changing. New apps, games, social media, and other technological advances are constantly being introduced, so teaching the same online safety lessons year-on-year isn't suitable. Keeping up with the latest trends is challenging, but remember that you work with the real experts! Ask your learners what's popular at the moment, how it works, and what safety measures they take when using that technology. Creating a safe space for them to share openly and honestly gives you the power to support them in making the right decisions to keep themselves safe and know how to handle new situations. Equally, your online safety curriculum should be reactive and allow you to address new issues as they arise.

Where is it relevant?

The conversation around online safety is no longer solely about online safety. It's about how learners handle their digital selves in all aspects of their development, and this requires both computing and non-computing

TIPS FOR INCORPORATING ONLINE SAFETY INTO THE SCHOOL DAY

- Read some online safety books such as *Chicken Clicking* (**helloworld.cc/chicken**) or *When Charlie McButton Lost Power* (**helloworld.cc/charlie**) to open up different conversations.
- In science, when learning about what humans need to stay healthy, could you extend this to a conversation about healthy and unhealthy behaviours with device use?
- When discussing stories about friendships in personal, social, health, and economic (PSHE) education classes or assemblies, do you include the differences between online friendships and offline ones? Opening up the conversations you are already having is a great way to encourage more honest discussion.
- ProjectEVOLVE has a range of activities to meet statements of the Education for a Connected World Framework (register to access these free resources at **helloworld.cc/evolve**). These range from discussion questions to vocabulary that inspires reflection. Why not use these to start or end your school day?

Your online safety curriculum is vital if your learners are to grow up safe and empowered

skills — for example, a learner's self-image and identity, managing relationships with friends, building a reputation, avoiding bullying and isolation, and having a healthy, balanced lifestyle. If we leave these skills to be developed solely in computing lessons, we not only misrepresent their importance, but we also wouldn't have a great deal of time left to teach anything else! Online safety has to be taught throughout the school day in various contexts, whether it's a casual conversation at break about online gaming with friends, or a social story about friends who were unkind either with or without technology.

In the computing classroom

After asking yourselves those three questions, there may still be some topics to be addressed in your computing lessons that are relevant to the content you are teaching. This is a great opportunity to use learners' developing technological skills while addressing online safety in a contextual way.

In the Teach Computing Curriculum (TCC), we teach learners about sharing personal information throughout the 'Creating media' units, as they consider what is and isn't OK to include in the artefacts they create (**helloworld.cc/tcc**). When learning about searching for media or information, learners explore managing information online and the issues they need to consider around copyright and ownership. Across all units, we encourage learners to manage their online accounts and think about the online reputation they are creating, as well as their rights to privacy and security. Each of these examples offers relevant opportunities to learn the skills of online safety and digital citizenship within the context of their computing units of work.

These TCC units are also linked to the Education for a Connected World Framework, created by the UK Council for Internet Safety (**helloworld.cc/connectedworld**). Even if you're not teaching in the UK, this framework is a great resource for understanding the breadth of online safety and digital citizenship. Although it includes recommended learning outcomes for learners aged 4 to 18, it doesn't recommend the most appropriate times and places to teach that content. For the TCC, we chose the most pertinent aspects to computing, but the rest is up to you.

Your online safety curriculum is vital for your learners if they are to grow up safe and empowered. Yet spending more time on topics relating to safety does not necessarily mean greater learning outcomes. Use these three questions as a starting point to review your online safety curriculum and to ensure that it is as prevalent as instructions about not running with scissors! (HW)

SWAY GRANTHAM

Sway is a senior learning manager at the Raspberry Pi Foundation. She leads a team developing computing resources for primary teachers (**@SwayGrantham**).

SAFEGUARDING IN ONLINE LESSONS

How do you organise live learning sessions that are both safe and help young people to learn? **Carrie Anne Philbin** investigates

Transitioning learning from a face-to-face interaction to online can sound straightforward, especially as we now live in a society where it's common to have access to devices and the internet. In a school environment, it is relatively easy to promote the welfare of young people and vulnerable adults and to keep them safe, thanks to well-established routines built on decades of learning. So if we find ourselves having to teach online, how do we promote the well-being of young people while they learn? Here are some tips that might help, based on feedback and ideas from brilliant educators and leading children's charities such as the UK's NSPCC (nspcc.org.uk), who have tested different approaches to hosting online sessions.

There are four areas to think about if you want to host online teaching sessions:

- Choosing the right technology
- Communicating with young people and parents
- Designing your session
- Child protection

Choosing the right technology

There are lots of different tools you could use to host live sessions, and they vary in their functionality, cost, and usability. When choosing a technology, think about how you intend to use it and how your intended audience will use it. Consider whether it allows private communication between you and young people, or between young people, as this could be a safeguarding risk. Use your school account and not a personal account when using online tools, and check the privacy settings. It's also a good idea to test the functionality of the technology with colleagues, perhaps by having a practice run of your session. They can stress-test any interactive features, and provide you with useful feedback to incorporate before you run it with students.

Another consideration is access. Does the technology you want to use require young people to have an online account? This may be an issue for learners below the age of 13. Do check your school e-safety policy, as it is likely that there is already guidance available on this issue.

SAFEGUARDING ONLINE GUIDES FOR TEACHERS

- NSPCC: undertaking remote teaching safely (**helloworld.cc/NSPCC**)
- GOV.UK: Education for a Connected World framework (**helloworld.cc/connectedworld**)
- GOV.UK: coronavirus safeguarding guidance (**helloworld.cc/safeguarding**)
- Childnet: teachers and professionals section (**helloworld.cc/childnet**)
- UK Safer Internet Centre: social media guides (**helloworld.cc/saferinternet**)

© Sushiman/stock.adobe.com

Communicating with young people and parents

Every organisation that provides activities for children and young people needs to get consent from a parent or carer for their child to participate. A well-written consent form will support your efforts to ensure parents, carers, and children understand the benefits and risks of online lessons, as well as providing written consent for children to take part. The NSPCC has an example consent form to help get you started (**helloworld.cc/consent**).

It's also a good idea to share a link to your online session in advance with parents, carers, and young people, as well as any instructions they will need for joining. You could also share what you are planning for your learners to work on, including links to any online projects or PDF files they may need. This will help your students to prepare for the session, and keep their adults informed about the learning you want them to experience.

> YOU HAVE THE SAME SAFEGUARDING RESPONSIBILITIES AS A TEACHER, WHETHER YOU'RE TEACHING ONLINE OR IN CLASS

Designing your session

As with any lesson, you should design the session structure and prepare your materials before you announce that you're going live online. If it is your first time using the online technology, I'd recommend having an introduction or starter activity that gives students the opportunity to play with the features. If there is a live comment stream, you might ask them to all say who they are and what they're hoping to learn in the session. I find that allowing this type of structured play reduces the opportunity for misusing the technology later.

You should also consider where you will present your session from. The NSPCC suggests you should be in "a neutral area where nothing personal or inappropriate can be seen or heard in the background".

I'd also advocate having another teacher or responsible adult acting as a teaching assistant during the lesson. They can moderate any feature misuse and keep an eye out for any safeguarding issues.

Child protection

Whether you are teaching online or in class, you have the same responsibilities as a teacher, and that means if you see or hear anything that worries you during the session, or a child discloses anything to you via email, you must disclose this to your child protection lead immediately. Make sure you have their contact details to hand and check your school's safeguarding and child protection policy and procedures. (HW)

CARRIE ANNE PHILBIN
Carrie Anne is director of educator support at the Raspberry Pi Foundation, and the host of Crash Course Computer Science and GeekGurlDiaries.

FEATURE

ACTIVITIES AND TOOLS TO BRING CYBERSECURITY TO LIFE

Rob Parker shares how his school teaches cybersecurity skills using real-life tools and projects, increasing engagement and excitement in the classroom

Teaching cybersecurity can be dull without hands-on practical experiences and exercises. Fortunately, my passion for cybersecurity and my over 20 years of experience as a data security engineer have helped me to understand how to motivate my students to acquire the real-world skills needed for success in industry. In this article, I'm going to present activities and tools you can use to link cybersecurity topics and skills to the world outside the classroom walls.

Projects with real-life consequences

I recently gave my students a taste of industry experience through a data security project about our internal IT infrastructure. Working alongside a team of remote penetration testers, I identified weaknesses in our own school system that could allow a data breach. We then set up various machines mimicking commonly known vulnerabilities, to demonstrate how a penetration tester could identify weaknesses and how they could be fixed. Running an activity such as this not only has the advantage of keeping our school network as secure as possible, but also helps students to understand the types of threat schools face, and how to put their learning into action.

This project particularly stressed the importance of keeping all operating systems up to date and using suitable security software, and the real-life consequences of not doing so. Students absolutely thrived during this project, as it allowed them to develop some basic skills that ethical hackers and penetration testers use in industry, as well as practical skills they can use in their everyday lives. It also allowed for topical classroom discussions in which students debated the advantages, disadvantages, difficulties, and ethics of such a project. Students are always interested to learn more from real-life case studies that help them relate personally to the importance of cybersecurity. Khanteepop Thaipradith, one of my students at Steam Labs, shared, "I learn ethical hacking to keep myself updated with the news and security. By going through the process of ethical hacking, I'm able to gain knowledge in order to use it to secure myself, as well as people around me, from being hacked."

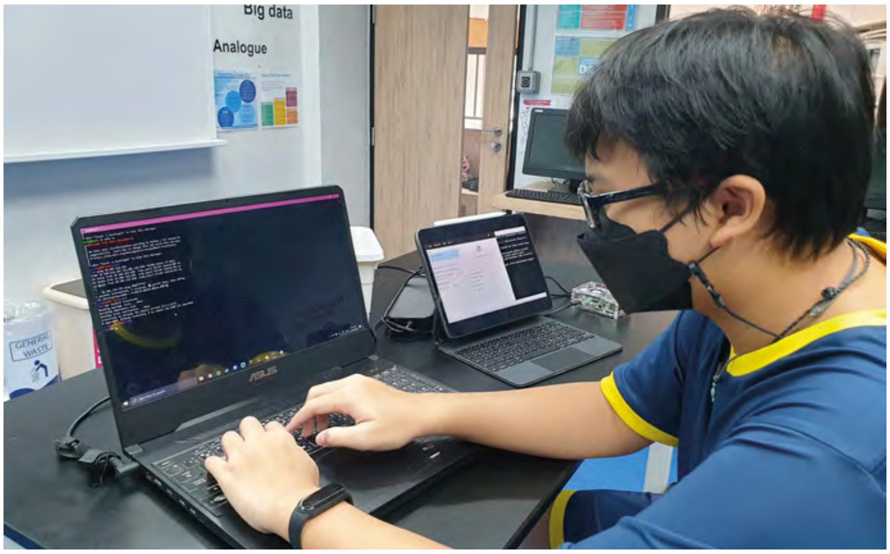
Keep students engaged with cybersecurity by using real-life tools and projects

Professional real-world tools

Another way of increasing engagement in cybersecurity teaching is through using tools that are actually used in industry. We use Raspberry Pis preloaded with Kali Linux, a system commonly used by industry specialists, to teach students about evolving threats in cybersecurity (**helloworld.cc/kalilinux**). Students learn the fundamental skills and steps needed to protect themselves against these threats, and the importance of ensuring that any data that is transmitted is encrypted. Using Kali Linux, students explore how to scan devices and sniff packets of data that are being transmitted. They learn how penetration testers identify whether particular ports are open on network devices using a TCP half-open port scan, and then look at what action a penetration tester would take if there were ports open that could be dangerous (you can learn more about how to do this at **helloworld.cc/sniffpacket**).

FURTHER RESOURCES

- **helloworld.cc/steamblog**: blog at Steam Labs (an online school that teaches robotics and ethical hacking), where I post free cybersecurity content for schools at the request of other global educators
- **helloworld.cc/robparker**: my LinkedIn profile; here, you can contact me for help with designing lessons and raise queries about how to integrate cybersecurity into the classroom



Another great way of putting cybersecurity into practice is by using Secure Shell. Using Raspberry Pi's command line, students learn how to access a device remotely by analysing the weaknesses in its security. The students take the perspective of the hacker, and search for areas of weakness. From there, we challenge them to think of ways of fixing the issues in the system. Any skilled ethical hacker or penetration tester has to be able to understand offensive security and think like a malicious hacker. Students are taught the ethics behind cybersecurity, and they thrive when learning about the skills that ethical hackers develop and use in higher education.

Similarly, we use Shodan (**shodan.io**), which can index every device connected to the internet, to demonstrate the importance of keeping devices that are connected to the internet up to date. We use it to explore how a hacker could potentially take advantage of a device using an outdated operating system. Wireshark (**wireshark.org**) is another powerful real-life tool you can use in cybersecurity lessons. It analyses network traffic and explores how encryption works, underlining the importance of having a strong password. Our students quickly discover how easy it is to crack a user's password using tools such as John the Ripper (**helloworld.cc/john**), and why systems block multiple password attempts when individuals use brute-force tools such as Hydra. Through these activities, we're developing their analytical and problem-solving skills, which they can transfer to everyday life and other parts of their education. Importantly, students finish these lessons with the know-how they need to help protect themselves and their families' devices from this type of hacking.

> ANY SKILLED ETHICAL HACKER HAS TO UNDERSTAND OFFENSIVE SECURITY AND THINK LIKE A MALICIOUS HACKER

VirtualBox (**virtualbox.org**) is another great tool, particularly if you're low on time and budget. It is a well-rounded, free, open-source piece of virtualisation software. This type of software allows you to install and use applications and operating systems other than those running on your computer. This means you can look at malicious files safely without infecting your computer, which can be a great starting point for applying cybersecurity knowledge and skills in the classroom. Raspberry Pi OS Lite is another good place to start. You can download a headless image and teach students how to access devices without a keyboard, mouse, or monitor.

If educators develop fun and exciting projects in cybersecurity, we will start to unlock our students' ability to excel, and help them to keep their own data and devices secure, as well as those of the organisations they work for in the future. (HW)

ROB PARKER
Rob is a computer science educator and the data protection officer at St Andrews International School, Bangkok. He is also a robotics and technology coach at Steam Labs (**steamlabs.co.th**), a specialist online school that teaches robotics, computer science, and ethical hacking. Rob is a certified Ethical Hacker with EC-Council.

KNOW THY ENEMY

Pretending to be the bad guys adds depth, understanding, and fun to cybersecurity classes, and helps us to become better good guys

Know thy enemy. This is one of the most famous tenets of warfare, according to the legendary Chinese general Sun Tzu. It's great advice on the battlefield, and its utility extends to the teaching of cybersecurity. In a way, cybersecurity is war: an ever-evolving conflict between those who want information and those who have that information. By the time I start teaching phishing, the majority of students have already been phished. If you want to beat the bad guys, I teach them, you have to walk a mile in their shoes to really understand them.

Walk a mile

When I introduce cybersecurity, I use as many real-world examples as possible. I keep every single phishing email I get in a folder, so that when we study phishing, we are studying primary sources. I pick out a few and we study each one as a class. We ask ourselves, what is this person trying to accomplish? How are they trying to accomplish it? What is their motivation? Understanding the offence improves our defence. Understanding the techniques phishers use helps us to identify them more quickly and accurately.

After studying these emails, it's time for students to show me what they've learnt, by phishing me. In doing so, they have to ask themselves the same questions: what am I trying to accomplish, and how? Some students' emails are more convincing than others, but they all make it clear that they know what a phishing attempt looks like. Most of them are funny, and occasionally, one of them actually fools me. Any student that does this gets extra credit and provides the entire class with another learning opportunity: it's a win-win situation.

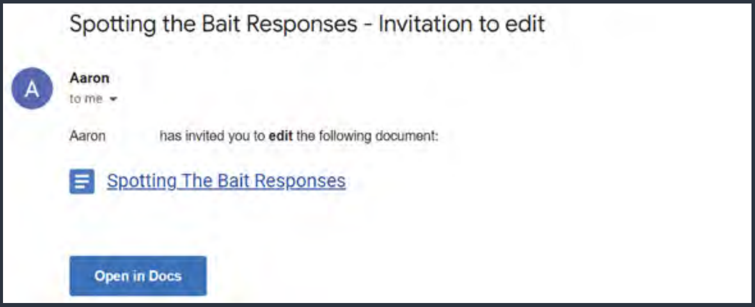
By studying and then creating real-life artefacts, we deepen our understanding of them. Being the bad guys makes us better good guys. So, the next time you study a threat, have your students emulate it. Have them write a clickbait title for a fake news article, or equate a spurious correlation with causation. In doing so, they gain a level of understanding that does not come with simply studying the subject. They understand their foe, and as Sun Tzu said, "If you know the enemy and know yourself, you need not fear the result of a hundred battles." (HW)

ZACH HUFFMAN

Zach is an upper-school computer science teacher at the Hun School of Princeton in New Jersey, USA. He is an avid gamer and has spent the last five years teaching computer science, from kindergarten all the way up to 12th grade (**@ltwheat**).

PHISHING CONTEST: THE WINNING ENTRY

Most real-life phishing emails try to alarm you into acting quickly and irrationally. Aaron, last year's winner, appealed to my lack of attention. I primarily use Google Docs for grading, and receive about 50 emails every week that all say "[Student] has invited you to edit the following document" with the little blue Google Docs icon. I click these without a second thought. Aaron knew that, and exploited it. This provided a meaningful teaching moment in class.



Spotting the Bait Responses - Invitation to edit

Aaron
to me

Aaron has invited you to **edit** the following document:

Spotting The Bait Responses

Open in Docs

KEEP CURIOUS AND CARRY ON HACKING

Ben Garside explores the meaning behind the word 'hacking' and makes the case for teaching students to hack

Over time, the meaning of the word 'hacking' has changed. Today, it often has pejorative connotations around bad things that bad people do on their computers. However, the term has much more positive origins, with its first recorded use coming from the Tech Model Railroad Club at the Massachusetts Institute of Technology (MIT) in the 1950s. Members of this club enjoyed taking apart model trains to discover how they worked, with the aim of enhancing them; they were 'hacking' the technology they were working with. In this article, I argue that hacking shouldn't be used as a dirty word, and that teaching students how to hack isn't necessarily a bad thing.

When I was a teacher, I often asked classes to open up old computers over a series of lessons about hardware. We would look at their components, see how they fit together, and discuss what we could do to improve the machines. This method of teaching students about hardware had more benefits than just visualising the abstract: without realising it, I'd been teaching my students how to hack. Just as Grace Hopper famously enjoyed dismantling clocks as a child to figure out how they worked, my students were learning how to be problem-solvers and how to fill the gaps in their knowledge. They were learning to hack, in the spirit of the original meaning of the term.

Developing students' problem-solving skills is no bad thing, but what if we were to take this a step further and also teach students how to hack in the more common, malicious sense of the word? When writing cybersecurity units for the Teach Computing Curriculum (**helloworld.cc/tcccyber1** and **helloworld.cc/tcccyber2**), this is exactly what we did. We made the conscious decision, in places, to go beyond the theory of how these attacks work and give students the opportunity to explore the code that makes them happen. We created activities in which learners use very unsophisticated code that could be used for a brute-force attack, as well as an activity in which students perform an SQL injection on a fake website.

We're not encouraging students to become criminals, of course, and we contextualise these activities with the ethics and legalities of these types of action. As Zach argues in the previous article, I believe that we're encouraging students to get a deeper understanding so that they can better protect themselves against real threats. I relate this approach to the karate principle. Although martial arts do teach people how to do harmful things to other human beings, a sensei is there to guide students in self-control and discipline. Relating this to hacking, students can easily go online and follow hacking tutorials, but at home, there is no sensei there to guide them about the rights and wrongs of their choices. As educators, we need to remember our role as sensei.

Let's keep teaching our students to be curious about technology, to want to know how something works, to be problem-solvers, and, therefore, to be hackers — just like the good people of MIT's Tech Model Railroad Club. (HW)

> WE DECIDED TO GO BEYOND THE THEORY AND GIVE STUDENTS THE CHANCE TO EXPLORE THE CODE BEHIND THESE ATTACKS

BEN GARSIDE
Ben is a learning manager for the Raspberry Pi Foundation. He has worked on the production of the Teach Computing Curriculum, and on online courses including the recently released *Introduction to Machine Learning and AI* (**@BenberryPi**).

EFFECTIVE USE OF TOOLS

The idea that learners start school as digital natives, competent in everything digital, is a myth. While some learners will have already developed confidence in using digital tools, this experience will vary greatly between students and will probably be limited to a narrow set of skills. In order to access and progress learning in all areas of a computing curriculum, learners need opportunities to develop skills such as using a keyboard and mouse, saving and organising their work, and more. This doesn't mean that we need to dedicate entire lessons to typing skills, but instead we need to provide opportunities to develop these skills alongside the broader concepts and skills within computing.

Initially, the focus for learners will be on developing the basic skills required to access learning, using a range of devices and simple software. As learners experience more devices, tools, and software, we can challenge them to see the commonalities between them, allowing them to learn to use new tools more quickly. They will then be able to be more selective in the tools they use for specific tasks, and will gradually become able to use specialist tools and software.

IN THIS SECTION, YOU WILL FIND:

- **Learning outcomes:** effective use of tools, in summary
- **What the research says:** the digital divide
- Digital skills for SEND learners
- Whole-school approach to digital skills
- Tablets and cross-curricular learning
- Young digital artists
- Logging on for lower primary

IN SUMMARY

EFFECTIVE USE OF TOOLS

Select and use appropriate hardware and software tools

STAGE 1

- Recognise the different applications of information technology in different contexts
- **Control a device using a touchscreen**
- Independently power up and sign in to computing equipment
- **Use a keyboard to enter and modify text, including the use of shift and backspace**
- Use a mouse to control a computer, including clicking, selecting, and dragging
- **Use a range of hardware, including cameras and programmable toys**
- Use a selection of online and offline software applications
- **Explain the purpose of different buttons and features of applications**
- Save work between sessions and reopen it
- **Explain how undo can be used to revert a change**

STAGE 2

- Identify the features and limitations of different devices
- **Describe how to use different tools successfully**
- Explain the uses of different communication and collaboration tools
- **Use a range of input and output devices to capture, manipulate, and share data and digital media**
- Connect digital devices and transfer files to and from them
- **Use appropriate communication and collaboration tools during a project**
- Find useful and suitable information online
- **Save work using useful and identifying filenames**
- Use cut, copy, and paste, as well as simple formatting tools
- **Apply existing IT experience and skills to new applications and contexts**

In the table below, you will find learning outcomes associated with the 'Effective use of tools' strand of the Raspberry Pi Foundation's computing taxonomy. These learning outcomes are illustrative of the kinds of knowledge and understanding that learners could develop in this area of computing. They are not prescriptive, but instead aim to illustrate the wide applications of the discipline.

These learning outcomes were originally developed to complement the English national curriculum for computing, and as such, stage 1 roughly corresponds to ages 5–7, stage 2 to ages 7–11, stage 3 to ages 11–14, stage 4 to ages 14–16, and stage 5 to ages 16–19.

STAGE 3	STAGE 4	STAGE 5
■ Describe how different hardware devices may be better suited to specific uses	■ Describe the features of specialist programming software such as IDEs	■ Use a range of specialist software in developing digital artefacts
■ **Describe how different software applications are suited to different purposes**	■ **Use collaborative revision tools to provide feedback and suggestions**	■ **Adapt to new software tools, combining past experience with available documentation and support**
■ Be familiar with a range of online communication and collaboration tools	■ Apply versioning to files to provide a version history	■ Use appropriate software tools to support planning, development, organisation, and collaboration
■ **Combine software tools to create digital products**	■ **Demonstrate responsibility for files through organisation and backup strategies**	■ **Use collaborative tools to connect and collaborate remotely with others**
■ Save and organise files within folders and create multiple versions where appropriate	■ Explain the purpose and benefits of templating and styling tools	
■ **Select hardware and software tools that are appropriate for a specific task**	■ **Use templating and styling tools to create consistency within digital products**	
■ Apply past experience of hardware and software to new devices and programs		

How many of the tasks you completed today could you have done without access to a digital device?

THE DIGITAL DIVIDE

STORY BY Ben Hall

How wide is your use of computers and digital devices? Think about your day so far: from the moment you got up to the moment you now find yourself reading this article, where and how have you interacted with computers? What are the purposes of your interactions with technology — work, social, play? Are these interactions all productive? And what else do you have planned for the day that will involve technology?

Now that you have reflected on your own interactions, what about those of others? Not everyone has access to digital devices. There has been a great deal of research in recent years into the digital divide, a term first used by psychologist Lloyd Morrisett. This is the gap between those who have and do not have access to computers and the internet. A 2019 study by Lloyds Bank identified that "11.9 million people (22%) do not have the Essential Digital Skills needed for day-to-day life in the UK" (**helloworld.cc/lloyds2019**). As the use of technology has expanded, the range of services available on devices has also increased, and consequently, so has the digital divide. Of the tasks you have completed using computers so far today, how many could you have completed if you had not been able to access some kind of digital device?

Causes of the digital divide

The causes of the digital divide are wide-ranging and complex. There are numerous papers offering ideas, including those from researchers Korupp and Szydlik (**helloworld.cc/korupp2005**). They analyse the causes of the digital divide through three lenses: human capital, which relates to people's education and their experiences with technology at work; family composition, which considers the make-up and income of a household; and social context, which considers an individual's generation, gender, ethnic background, and geographical location. Within these groups there are complex factors that can either exacerbate or mitigate the divide. Whatever the cause, the impact is that these people or groups are excluded from some aspects of society.

The digital divide came into sharp focus during the coronavirus pandemic. The need for people to isolate from each other, and the expansion of online services, meant that the pace of change increased hugely. Schoolchildren found themselves in home learning environments, needing laptops or tablets and, crucially, bandwidth, to access their learning. Such a scenario was unthinkable before the pandemic. Some children and families adapted quickly and easily, using devices they already had and dedicating spaces in their homes to school-based activities. Others were less fortunate, coping with very limited space and having in many cases to share devices unsuitable for learning, such as mobile phones, between multiple siblings. Researchers Holmes and Burgess highlight that only 51 percent of the poorest households have internet access, compared to 99 percent of the richest (**helloworld.cc/holmesburgess**); it is very clear that the pandemic has exacerbated existing inequalities.

> IT IS CLEAR THAT THE PANDEMIC HAS EXACERBATED INEQUALITIES BETWEEN THE RICHEST AND POOREST HOUSEHOLDS

Looking through Korupp and Szydlik's social context lens, older people have also been badly affected by the pandemic and the acceleration of the digital divide. Services such as booking a doctor's appointment or ordering a prescription moved online, and the pace of change left many behind, with inadequate support provided to help this group come to terms with new ways of doing things.

Digital natives

Are any groups immune to the digital divide? You may have come across the term 'digital native', a term first introduced by researcher Marc Prensky in 2001 (**helloworld.cc/prensky2001**). This term is often used to describe young people who have been brought up using computers and the internet from a very young age. Unfortuantely, articles such as Prensky's perpetuated a widely held belief that if you can use a tablet or phone to access content online, you are somehow a digital native and can use these skills across many platforms and technologies — therefore, this group surely cannot be victims of the digital divide?

These beliefs have now been challenged. In a 2019 study, researcher Scolari focused on media literacy, breaking down the term into ten 'new literacies' (**helloworld.cc/scolari2019**). He found that the term digital native has "more problems than advantages". Not all young people, for example, have access to devices, and much of the behaviour of these so-called digital natives is focused on the consumption of digital content, such as watching something online or playing with a device. This does not involve using digital tools to their full extent; consumers are not producing anything by doing this, or thinking about how they could make best use of the tools they have at their disposal. In fact, labelling people as digital natives can actually widen the digital divide. It peddles the false assumption that the skills associated with the effective use of tools do not need to be taught to learners, which could lead to a life surrounded by technology they can never fully understand or exploit.

Bridging the digital divide

So what can we do? We cannot ignore the problem. The growth in the use of computers and the internet is unstoppable, but that does not mean the digital divide also needs to widen. More work needs to be done to ensure that everyone has access to digital tools as and when they need them, as well as access to opportunities to help them develop their skills. While computers are still expensive, there are tools available that can provide access to key services at a reasonably low cost. Raspberry Pi is a great example — for less than $50, you can buy a fully functional personal computer that will plug straight into most modern TVs. From the start of the pandemic to the end of 2021, the Raspberry Pi Learn at Home initiative distributed nearly 6000 fully functional computer sets to educationally disadvantaged young people in the UK. The impact was immediate: young people were more engaged with learning; parents reported positive changes in their children's attitude and behaviour; and youth and social workers deepened their relationships with families, enabling them to provide better support (**helloworld.cc/RPFreview2021**).

There have been a number of other similar initiatives advocating the idea of one laptop per child (OLPC). However, it's important to stress that equipment alone will not necessarily deliver improvements. Researchers Thapa and Sein studied an OLPC scheme in Nepal that had been widely regarded as a success. They found that it wasn't the technology per se that brought quality education to these schools, but rather, it was the ecosystem around the deployment of these devices that had a more significant role in the success story (**helloworld.cc/thapa2018**). Similarly, the Learn at Home initiative has been backed by significant support, to ensure that students were able to get the most from the devices they were provided with.

While the pandemic widened the digital divide, it also drew attention to it. With the issue now more in the spotlight, there are opportunities for more research and more initiatives to broaden digital inclusion and ensure that digital tools can be used effectively by all. (HW)

FURTHER READING

- ✓ Prensky, M. (2001). Digital Natives, Digital Immigrants, Part 1. *On The Horizon. 9*(5), 1-6. **helloworld.cc/prensky2001**
- ✓ Thapa, D., & Sein, M. K. (2018). An ecological model of bridging the digital divide in education: A case study of OLPC deployment in Nepal. *The Electronic Journal of Information Systems in Developing Countries. 84*(2). **helloworld.cc/thapa2018**
- ✓ Korupp, S. E. (2005). Causes and Trends of the Digital Divide. *European Sociological Review (ESR). 21*(4), 409-422. **helloworld.cc/korupp2005**

FEATURE

KEY DIGITAL SKILLS FOR YOUNG PEOPLE WITH SEND

Catherine Elliott shares the importance of teaching key digital and digital literacy skills, and looks at how you can build them with your students

Computing curricula generally cover a wide range of skills, concepts, and knowledge, and much of the focus is often on creative projects, programming, and abstract computer science concepts. There is a powerful argument, however, for ensuring that we teach the basic skills well in the first instance, with a particular benefit for young people with special educational needs and disabilities (SEND).

Basic digital skills are any skills that are required to access and use a computer effectively. The concept of digital natives has been widely discredited, but many teachers believe that young people just 'get' computers and can use them with ease, as they are more confident users. However, although children may be experts at swiping and accessing content on certain devices, such as tablets, they often lack keyboard and mouse skills, and the understanding of the basics of an operating system.

Basic digital literacy is equally important. The skills and knowledge required to communicate effectively and to use current and emerging technologies are essential if students are to remain safe and act appropriately online. Young people with additional learning needs and disabilities can be among the most vulnerable in terms of online risks and behaviours.

Why is developing fluency in basic digital skills and literacy important?

1. It reduces cognitive load

Cognitive load relates to the number of items a person can hold in their working memory. For many younger pupils and students with SEND, basic skills such as logging on and opening files are not practised enough to be moved to long-term memory, and therefore rely on working memory. This results in the frustrating position of a child spending much of a lesson simply accessing a computer and finding work. Once these actions become fluent, learners can concentrate on the content of the lesson more effectively.

2. It increases confidence in using technology

Once learners become fluent in using the computer in basic ways, their confidence will develop. This will help to increase their motivation and contribute to a feeling of achievement.

3. It enables learners to use assistive technologies effectively and make simple modifications to content to support their learning

Mainstream applications and devices now have a greater amount of assistive technology built in — for example, Immersive Reader in Microsoft 365, Voice Typing in Google Docs, and Speak Screen on the iPad. If we can teach young people how to use these options independently as part of computing lessons, it will help them to become more effective in their learning. Similarly, highlighting how to increase the size of text or change the background colour in documents will allow learners to modify digital documents to be more accessible.

4. Digital skills open up greater opportunities for employment

There are few jobs where digital skills are not required, and even the application process generally requires the use of some elements of technology. For students who are not taking an IT or computer science qualification, schools need to consider how to teach key employability skills, such as sending emails and searching for information online. A functional skills qualification in IT would greatly benefit some learners with SEND in preparing them for their next steps in education or employment.

5. It helps students use technology safely and responsibly

Young people need to be taught about the risks of online technologies, how to act appropriately online, and where to go for help with safely accessing key services. They need the same opportunities for learning, entertainment, and shopping as their peers.

ASSISTIVE TECHNOLOGY IN MAINSTREAM APPLICATIONS

- Support weaker readers with the Immersive Reader tool, built into Microsoft 365 products and the Edge browser, and as an extension in Chrome (see **helloworld.cc/immersivereader** for more info)
- Enable Speak Selection in the Accessibility settings on the iPad to enable students to listen to any selected text
- Try the Voice Typing option in the Tools toolbar in Google Docs to allow learners to convert speech to text; there is also a dictation function via the keyboard on iPad and Android tablets, and in Microsoft Word online

How can we teach key skills effectively?

- Develop fluency through routine. For example, pupils open and save work from the same folder each lesson, and have a routine for logging on when they enter the ICT suite or access a laptop. Share these routines with the other teachers these learners work with.
- Provide support materials for learners for habitual actions — create a set of simple instructions for each step, with image support. For example, log on, open an application, save work, and take a photo on the iPad.
- Provide lots of opportunities for repetition to consolidate learning — time spent repeating an action multiple times, or practising keyboard and mouse skills, is worthwhile for helping key skills to become fluent early in the year. Consider how to combine this with other meaningful tasks in the classroom, or to assist with other learning goals (for example, typing practice to support spelling).
- Teach learners how to use assistive technologies built into the mainstream tools that are available at school or at home through a learning platform. There are a number of tools to allow students to dictate rather than type, have text read to them, or add subtitles to videos. Where possible, share these with parents to use at home.
- Show students how to adapt digital content to make it more accessible. A great benefit of the pandemic has been that we have been able to provide lesson content for students to access in their own time, so they can revisit and consolidate their learning. If this is in an editable format (such as Word, PowerPoint, or Google Docs), learners can change the background colour, increase the font size, choose a more readable typeface, and increase line spacing to make it more readable. If you are providing videos to watch, enable the use of closed captions.
- Model safe and responsible use of technology in the classroom, and make your actions explicit through commentary and discussion.
- When teaching digital literacy and online safety, make sure you discuss risks and behaviour in several different situations and contexts, as some young people with SEND struggle to generalise their knowledge. Issues also need explaining explicitly for those students who can't infer risk from subtle cues.

Computing often doesn't receive enough space in the curriculum. However, an investment of time and effort in developing key digital and digital literacy skills early in a child's computing journey will pay dividends later in increased confidence, fluency, and access to learning for all of your students. (HW)

CATHERINE ELLIOTT
Catherine is the SEND lead for the Sheffield eLearning Service (**sheffieldclc.net**), and she works on ways to make computing accessible to all learners. She is a member of the CAS Include working group, and leads the SEND Virtual and the Sheffield and South Yorkshire Secondary CAS Communities (**@catherinelliott**).

A WHOLE-SCHOOL APPROACH TO DIGITAL SKILLS

Claire Buckler shares the benefits of embedding digital skills across the curriculum

Like most schools, at Devonport High School for Boys (DHSB), we want to ensure that all of our students are equipped with the digital skills they need to thrive in the world. We have been working hard to ensure that even those students who don't choose to take computer science are technologically literate. Across the school, technology has a strong presence. We operate a bring-your-own-device policy, which will shortly be replaced with a one-student-to-one-Chromebook scheme. Most departments have Chromebooks or iPads, and we encourage students to use mobiles phones as learning devices. As a Google Reference School, we use Google tools in creative and innovative ways across the whole of our curriculum. This healthy relationship with technology is essential for our students, to ensure that they are data-aware and can use digital tools to communicate and collaborate — vital skills for being digitally literate after leaving school. We offer a coding club here twice a week, which is full of students who don't take computer science as an option but still see the benefits of learning to code.

The benefit of support

All of this comes from having a school that sees digital literacy as an essential skill and is happy to provide the time and resources to get all staff and students on board. My role is a great example of this: I am the director of the Learning Commons, a library and digital breakout space. We have 60 Chromebooks, which are available at break times, and an innovative learning space. The Learning Commons is often booked out by teachers who enjoy the freedom that the space offers compared to a traditional classroom. Our assistant head, Nick Berryman, will bring his business students in when they need to access technology. He says, "Digital literacy is an important skill, and at DHSB we have embraced this. The Learning Commons inspires the future generation to be creative and learn the skills needed to adapt." Google Expeditions is currently a favourite app among teachers and students, who can use our class set of Google Cardboards — the affordable VR headsets.

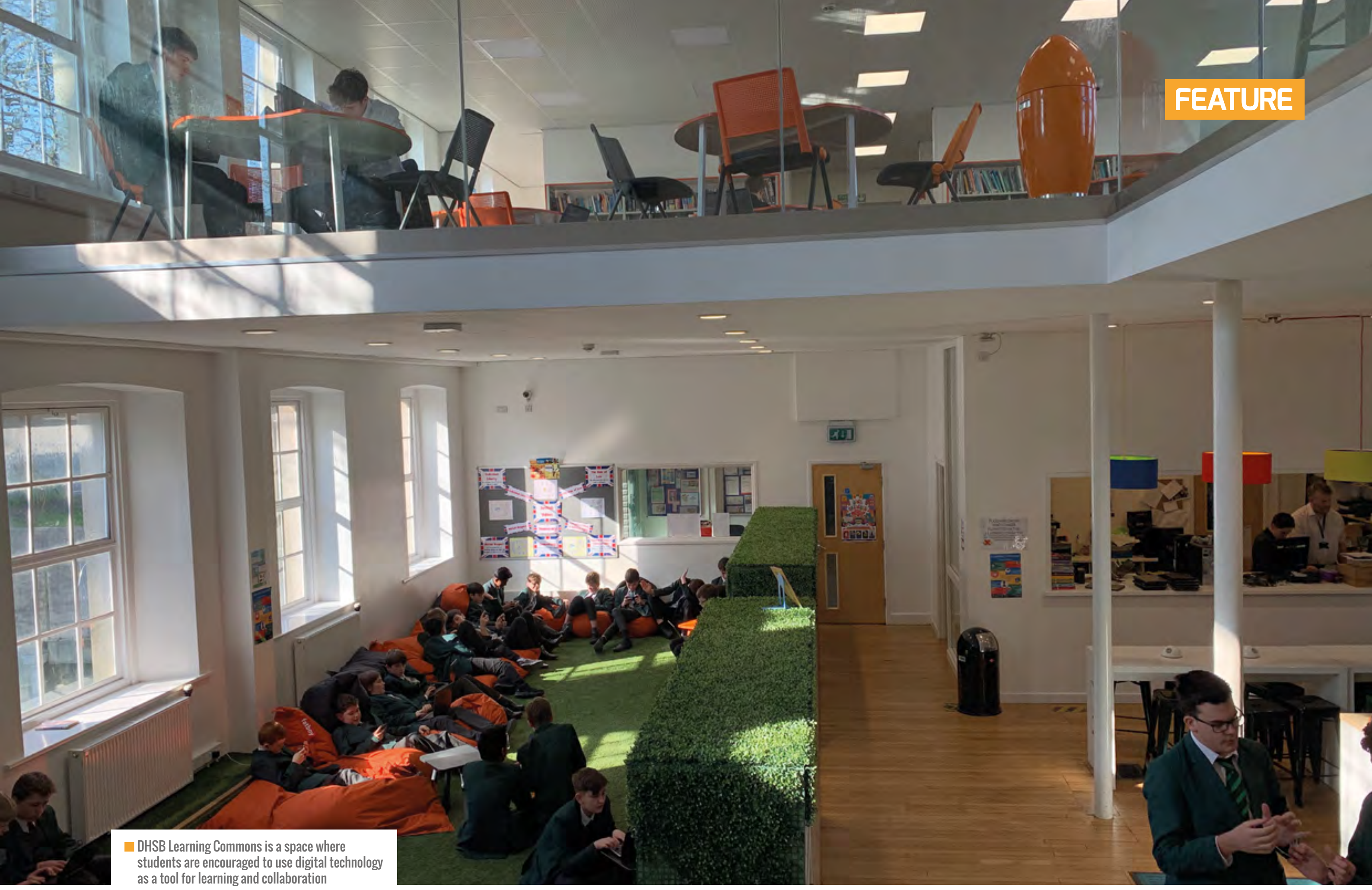
Another way we involve students in technology is with our Digital Leader scheme. Students take the roles of IT helpdesk staff, and we assign technical support tickets to them. They are an amazing resource for the school, saving time and allowing technical staff to concentrate on infrastructure issues. Students who are successful in their bid to be part of the Digital Leader scheme take the Google *Applied Digital Skills* course (**helloworld.cc/digitalskillscourse**). They are available to run theatre lighting, set up for assemblies, lead Code Club sessions, and produce graphics and informatics for school departments.

We currently have over 30 Digital Leaders, with a healthy waiting list. Some of the team are also e-safety ambassadors and run relevant workshops. The students involved show fantastic problem-solving and creative thinking skills. But the most amazing thing about these students, in my opinion, is that very few of them are taking computer science as an elective at GCSE. Clearly, they have a great relationship with technology. One of our Digital Leaders,

BECOMING A GOOGLE REFERENCE SCHOOL

DAN ROBERTS, HEAD TEACHER AT DEVONPORT HIGH SCHOOL FOR BOYS

Devonport High School for Boys was approached to be a Google Reference School and we opted to use Google for Education to reduce costs and improve productivity and efficiency. More importantly, we value how the apps empower young people to collaborate, rather than more traditional virtual learning environments which limit collaboration and learning.


DHSB Learning Commons is a space where students are encouraged to use digital technology as a tool for learning and collaboration

George, told me he feels he has improved his digital skills by being part of the team, and this had enabled him to take different elective subjects at GCSE.

No time for complacency

With constant access to technology, we need to be mindful that our students don't fall into the trap of becoming passive users, but stay curious and enthusiastic about how technology works. We need to ensure that we still encourage students who do not opt to study computer science to find technology interesting and relevant. The next step for our school is to implement a maker space. Our maker space will be part of the Learning Commons, and students will be encouraged to experiment there at break times and after school. Rather than being shoehorned into one subject, any teacher will be able to use the space to create and innovate across the curriculum. Part of our culture here is to inspire an innovative and entrepreneurial spirit in all our students. Maker spaces achieve this by encouraging students to design, experiment, and build in a hands-on environment, without being required to meet any specified outcome, and the best thing about it is it will be led, of course, by our Digital Leaders.

"STUDENTS IN OUR DIGITAL LEADER SCHEME ACT AS OUR IT HELPDESK AND WE ASSIGN TECHNICAL SUPPORT TICKETS TO THEM

Getting started

While having a supportive senior leadership team has been key to our success, there are many small wins available. Setting up a Digital Leader scheme may seem labour-intensive, but we have quickly seen the benefits. Offering a Code Club is simple — the **codeclub.org** resources don't need any planning. A maker space may seem a lot of work, or expensive, but could be as simple as a few Raspberry Pis or micro:bits, with some electrical components, most of which are reasonably priced. All of these could encourage more students, outside of those who choose to study computing, to be inspired by technology. (HW)

CLAIRE BUCKLER

Claire is director of the Learning Commons and a teacher of computer science at Devonport High School for Boys in Plymouth, UK. She is a Level 2 Google Educator and Community Leader/Trainer for Computing at School (**@clairegowland**).

USING TABLETS TO ENHANCE CROSS-CURRICULAR LEARNING

Charlotte Spenceley shares her experience of incorporating tablets into her teaching approach, and how this has benefited both her and her primary pupils

I am fortunate to work in a school where significant investments have been made to promote and develop the computing curriculum — and one key benefit is that every pupil aged seven to eleven has their own tablet.

Nearly two years ago, I was given the opportunity to work with an experienced computing specialist to develop a cross-curricular approach to using tablets in the classroom. This opportunity came about through fortunate timetabling and as a result of my reputation for being interested in using technology with my students — I'm one of a small number of people who know how to get the interactive display and sound in the school hall to work at the same time! The specialist, who was known to our head teacher, was employed at the school to lead the computing curriculum, provide our pupils with stimulating opportunities, and deliver continuing professional development for staff.

INTRODUCING TABLETS TO YOUR PUPILS

- Let children explore the apps
- Teach them what key symbols mean; for example, press + to add something
- Introduce one skill at a time

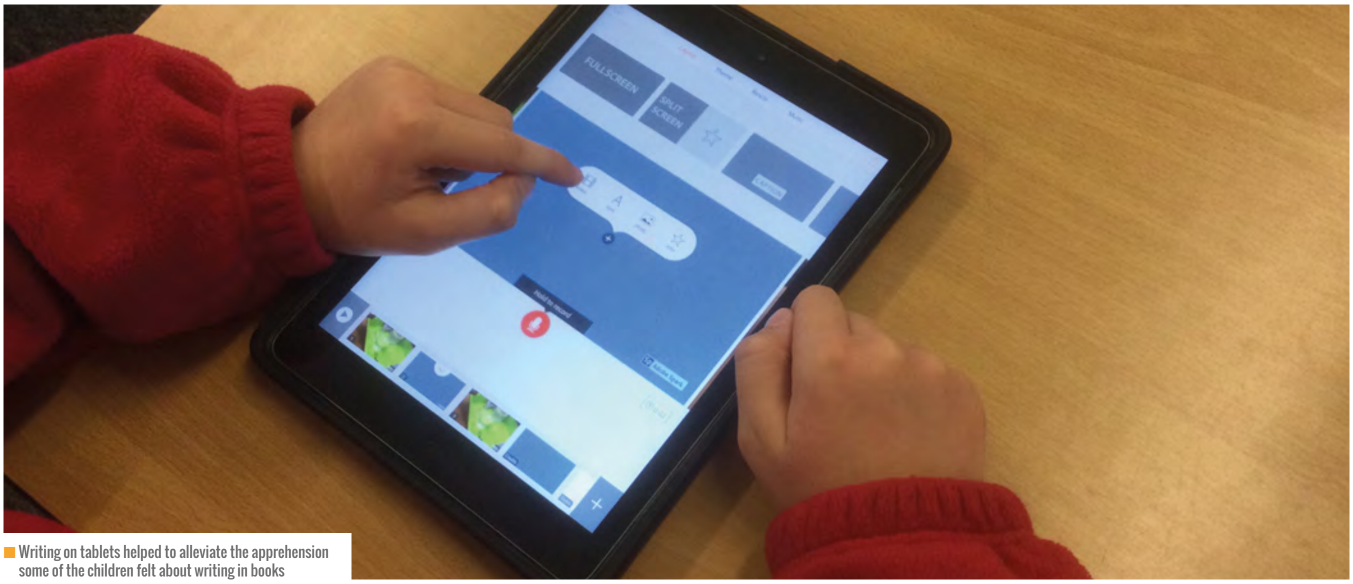
When implementing the use of tablets in the classroom, our aims were for pupils to develop key literacy skills, explore new routes for creativity, and learn how to record and evaluate their work.

Because of these investments in developing the computing curriculum, I have learnt more about using technology in the classroom in the past 18 months than I have in almost ten years of teaching. Prior to this, my experience of computer science was very limited and I had scarcely been exposed to computing or technology in the classroom. Tablets were my starting point, and I had one afternoon a week with our specialist, and a science curriculum to adapt around the topic of rocks. We began our journey with this topic for timetabling reasons, but quickly adapted the lessons learnt to other areas of the curriculum, such as history and English.

Getting started with the tablets

Our journey began with eliciting the pupils' understanding and curiosity by using the PicCollage app to write questions over a photograph of a rock. Straight away, my students and I encountered problems: we couldn't get the camera covers off the cases; we didn't know how to capitalise letters; and we were afraid to press any buttons

Using Adobe Spark Video, we were able to bring to life an investigation into the permeability of rocks



Writing on tablets helped to alleviate the apprehension some of the children felt about writing in books

> ONE OF THE MOST POSITIVE CHANGES IN MY PUPILS IS THE ABILITY TO REFLECT ON THEIR OWN EVOLVING KNOWLEDGE

for fear of "destroying the tablets" (to use the children's words). I questioned myself throughout: was the time used to model every tap worth it? Was every child making progress from their individual starting point? What was the difference between this and the tried and tested strategies?

Despite these initial hurdles, the dialogue created through this activity allowed for thoughtful, memorable, and reflective learning among the group. Writing on the tablets also helped to alleviate the apprehension that some of the children felt about writing in exercise books. Misconceptions were identified, understanding was assessed, and I was able to reflect on my teaching. As with most things in a school environment, I learnt that patience needed to be embraced, and that the quality of participation, confidence, and problem-solving skills witnessed in one afternoon justified the time needed to embed routines. Most importantly, I learnt that children need to be taught how to use technology in a meaningful way, and not to assume that all young people are digital natives — a term often used to describe the generation that has grown up in the era of technology and the internet.

Growing more confident

We were eager to build on our initial exercise, and our subsequent activities explored different methods of recording. A popular app with my class is Balloon Stickies Plus, an application that allows you to add speech bubbles to photos. Not only do they like the name, they also enjoy being able to choose between typing and voice recording, and the freedom to present their work the way they want to. I have found that taking away the structure of an exercise book has allowed children to focus on the content and creativity of their learning.

One of the most positive changes I've seen in my pupils is their ability to reflect on their own evolving knowledge. Using Adobe Spark Video, we were able to bring to life an investigation on the permeability of rocks. We used technology to look back at prior learning, and this helped to predict and justify the results. As each experiment was conducted, the tablets were used to record the results visually. Spark Video supported our observations, reasoning, and conclusions by allowing us to explain each step verbally and annotate the video with additional information. Having to record experiments by written methods can sometimes hinder engagement and enthusiasm. However, by patiently developing the skills needed to use tablets as an effective learning tool, children can grow more confident with taking risks, and become more engaged, reflective, and digitally literate learners.

Overcoming barriers

I am still working to overcome some barriers: balancing book work with technology; figuring out how to show constructive marking and give feedback on the tablets; and planning the next steps in embedding the cross-curricular use of technology in my setting. However, my pupils are becoming "confident", "reflective", and "enthusiastic" learners (to use their own words again) and I believe that this shows the strong impact that tablets can have when used as a learning tool in the primary classroom. (HW)

CHARLOTTE SPENCELEY
Charlotte is an upper-primary teacher and school council lead at Giffard Park Primary School in the UK. She is also a newly qualified teacher mentor and is the health and well-being champion at her school.

CREATING DIGITAL ARTISTS: YOUNG PHOTOGRAPHERS

Creating future artists and photographers is much more than just point and shoot

For the past three years, I've had the pleasure of running a photography club for a small group of children. It's given me the opportunity to share one of my own passions with the pupils, as well as helping them to get creative more often. In my opinion, too much of primary school life is taken up by English and maths! We need to give pupils the chance to let their creativity flow, and create something unique and memorable.

Each week, the club meets up and spends an hour playing with a set of cameras in and around our school. I set them a challenge or a theme each week, and we head off to see what we can snap. It might be as simple as giving them a title such as 'Stone' or 'Shadows' and seeing what they come up with. Other weeks, we might go on quick trips, or maybe look at the work of a professional photographer. The teaching comes from talking about photography concepts during each of these activities. We talk about lighting, composition, the rule of thirds, and what they think makes a good photograph.

Sharing is key

There's no point in the pupils taking all these wonderful pictures if they are never seen again! Our club has a few different ways of sharing their beautiful work. First, at the start of the year, they're given a large art pad to be used as their portfolio. This is for them to keep and use. It's not a book that gets marked, but a pad that gets admired. It should be a reflection of their work and a space for them to show off their talent.

The second way is our own little blog space. The idea is for the pupils to keep it up to date and share their work. It looks great and is an excellent way for them to showcase their photos. The trick with this is to find a blogging platform with a healthy storage allowance, otherwise you'll fill it up pretty quickly. This has also been a great way for us to share the work with students' parents. I can quickly send out the link in an email so they can see the great work their kids do in the club.

The last method is a favourite of the pupils. Taking a wall in one of the main school corridors, we've created our very own Photography Club Gallery. They get to pick their favourite snaps (some of them would not be my choice!) and display them in frames, labelled with their names and the camera used. All I did was buy the cheapest frames I could find at IKEA, and they love the gallery. We have a three-month policy on changing the photos, just to make sure the gallery stays fresh and new. The pupils have even been commissioned to take some shots for the staffroom walls by the deputy head teacher! This is such a positive experience for the kids and is hopefully something they'll remember for a long time.

Simple equipment

When getting started and looking at buying equipment for a club like this, it's important to remember that the type of camera used really doesn't matter. Honestly!

We started off with some simple point-and-shoots: bog-standard, cheap, and cheerful. The trick was to get the kids thinking about how they could use them in different and interesting ways, and how they could make sure the subject in the photograph was captivating. After a few months, I managed to beg, steal, and borrow from the head to get some Sony A5000s. These cameras are great. They have all the power of a DSLR, but they're small enough for little hands. These were a bit pricey and are fairly delicate — a scenario that will fill all primary-school teachers with dread. So with the little funding I had left, I bought a few 'rough and tough' cameras. I chose the Ricoh WG-30, and I can't recommend these cameras enough. They're waterproof, shockproof, and even have digital microscope capability. They really

The DSLR pro photographers!

On a cold and frosty morning, use the macro setting for some interesting close-ups

Use the slow shutter speed on a DSLR to paint with light

A simple glass ball can create magical images

Broken equipment is given new life in the hands of inquisitive minds

stopped me from worrying when a child was running down the mud track in the middle of the Yorkshire Dales!

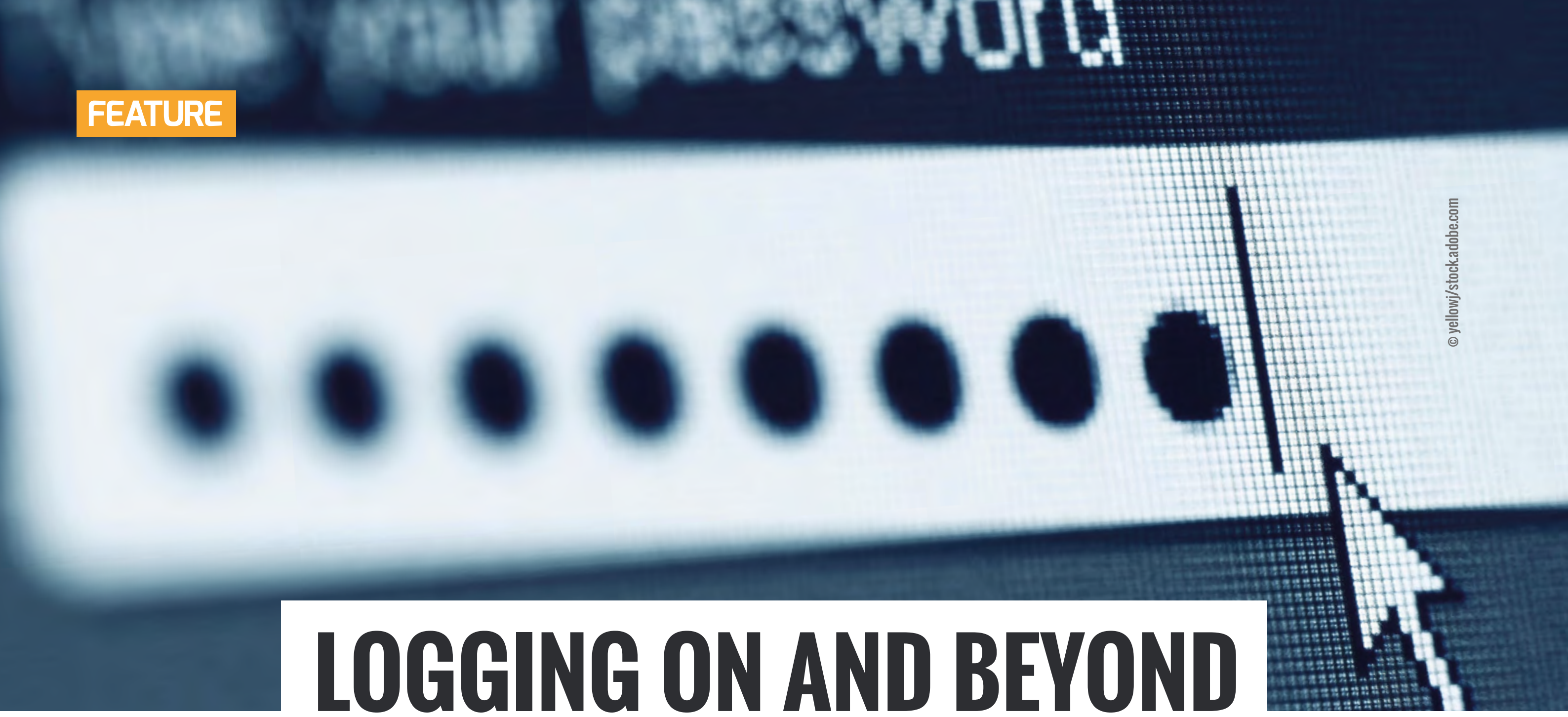
Allowing the children to investigate and explore with the cameras has been one of my drives for the club. One way I've done this is by collecting bits and pieces for us to use with the cameras. I've been to camera shops and bought old, broken, and scrap SLR lenses. They're great for looking at how a camera works, but also for holding up to the lens of your cheap camera and taking some interesting snaps. I've also added a few small torches for painting with light. Do a few Google searches to find out how you can achieve this; it's incredible. Using a slow shutter speed, you can paint using the light of a torch. So cool! My personal favourite piece of equipment, though, is a glass ball. Pick one up on eBay for little money, and get ready to take some unbelievable pictures. They make even the most novice photographer look like a pro.

Three years on...

Having worked with the same small group of children for such a long time, I've seen them develop right before me. The club has seen them change into future artists. It has instilled in them a sense of aesthetic. It has given them the opportunity to create something with the sole purpose of looking good. I'm not sure how often we get to do that in a primary school anymore, with all the pressure of tests. My biggest hope is that it's passed on a love and enthusiasm for photography in the pupils, and that it's something they'll remember doing. I can't wait to see some of their portfolios in future life. Fingers crossed! (HW)

MATTHEW MOORE
Matthew is a computing specialist at a primary school in Bradford, UK. He is also a CAS Master Teacher and Hub Leader, as well as a Raspberry Pi Certified Educator.

FEATURE



LOGGING ON AND BEYOND

Sway Grantham shares tips on how to teach children under seven to log on to school computers independently

Just like blowing their noses or tying their shoelaces, logging on to school computers is a skill young primary-school students need to learn. However, many children can find logging on by themselves quite challenging. Here are some top tips to help you get your youngest learners logging on to devices independently.

Simple usernames and passwords

Consider what is making it hard for the learners to log on. If it's unrealistic usernames and passwords, you should speak to the IT manager, or whoever controls your logins, and make these appropriate for the age and ability of the children. Their first name and a '1' for Year 1, and a three-letter password, is more than enough for this age group. However, you should avoid them all having the same password, as we do not want to advocate this, even with our youngest learners.

Check and build the foundations

There is a range of foundational skills that children need before they can log on independently, including turning on and shutting down computers safely, using a keyboard, and using a mouse to click in a box. Do your students have these skills already? If they don't, it's worth spending some time on developing them, so that logging on isn't an overwhelming task.

Teach it explicitly

If you plan a lesson in which the outcome is that learners have logged on successfully, you can break down each step and take your time, without feeling the pressure of moving on to other lesson content. For those learners who still struggle, a visual prompt is often useful — you can hand them a chart showing each of the steps, to encourage them to continue independently.

Peer support

Can the children who have mastered the skill support those who haven't, until they become more independent? I would often set table challenges for everyone on one table to get logged on, awarding school points for the fastest table. This encouraged the more capable children to support those who were struggling, reducing the time pull on the teacher, and also allowing the learners to learn from their peers. I also had a rule that the only person allowed to touch the computer was the person who was using it, so the helpers could tell them what to do, but not do it for them!

Don't let it be a barrier

If, after trying all of the above, you still have some learners who are struggling, they should still be able to access the computing curriculum with support. Just like if a child was unable to dress themselves for PE, you would eventually intervene, but you might set them a personal target to work on that specific skill until they got there. We don't want to impact the children's attitude to computing because they struggle with logging on.

Logging on is a skill that many adults take for granted, but it is a skill in itself. We need to give it the amount of time it deserves, so that we can get the majority of our learners to do it independently as soon as possible. This not only makes our lives easier, but it also broadens the learners' opportunities for using technology across the curriculum in the future. (HW)

SWAY GRANTHAM
Sway is a senior learning manager at the Raspberry Pi Foundation, where she leads a team developing computing resources for primary teachers (**@SwayGrantham**).

NOTES

"HELLO, WORLD!"

Everything you need to know about our computing and digital making magazine for educators

Q WHAT IS HELLO WORLD?

A Hello World is a magazine and accompanying podcast for computing and digital making educators. Written by educators, for educators, the magazine is designed as a platform to help you find inspiration, share experiences, and learn from each other.

Q WHO MAKES HELLO WORLD?

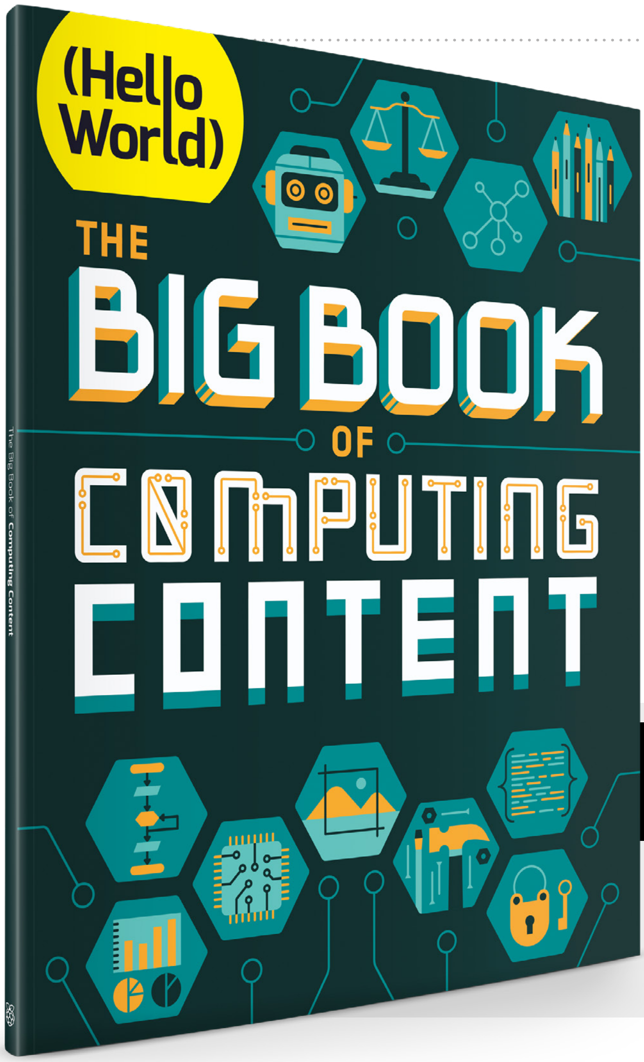
A The magazine and the accompanying podcast are produced by the Raspberry Pi Foundation.

Q WHY DID WE MAKE IT?

A There's growing momentum behind the idea of putting computing and digital making at the heart of modern education, and we feel there's a need to do more to connect with and support educators, both inside and outside the classroom.

Q WHEN IS IT AVAILABLE?

A Your 100-page magazine is available three times per year — keep an eye out for special editions too! Check out our podcast at **helloworld.cc/podcast** to get more great Hello World content between issues.



IT'S FREE!

Hello World is free now and forever as a Creative Commons PDF download. You can download every issue from **helloworld.cc**. Visit the site to see if you're entitled to a free print edition, too.

WANT TO GET INVOLVED?

There are numerous ways for you to get involved with the magazine — here are just a handful of ideas to get you started

● **Give us feedback**
Help us make your magazine better — your feedback is greatly appreciated.

● **Ask us a question**
Do you have a question you'd like to share? We'll feature your thoughts and ideas.

● **Tell us your story**
Have you had a success (or failure) you think the community would benefit from hearing about?

● **Write for the magazine**
Do you have an interesting article idea? Visit **helloworld.cc/writeforus** to submit your idea.

GET IN TOUCH

Want to talk? You can reach us at:
contact@helloworld.cc

FIND US ONLINE

www.helloworld.cc

@HelloWorld_Edu

fb.com/HelloWorldEduMag

NEXT ISSUE OUT
JANUARY 2023
THEME: COMPUTER SYSTEMS & NETWORKS

(Hello World)

helloworld.cc