(Hello World)

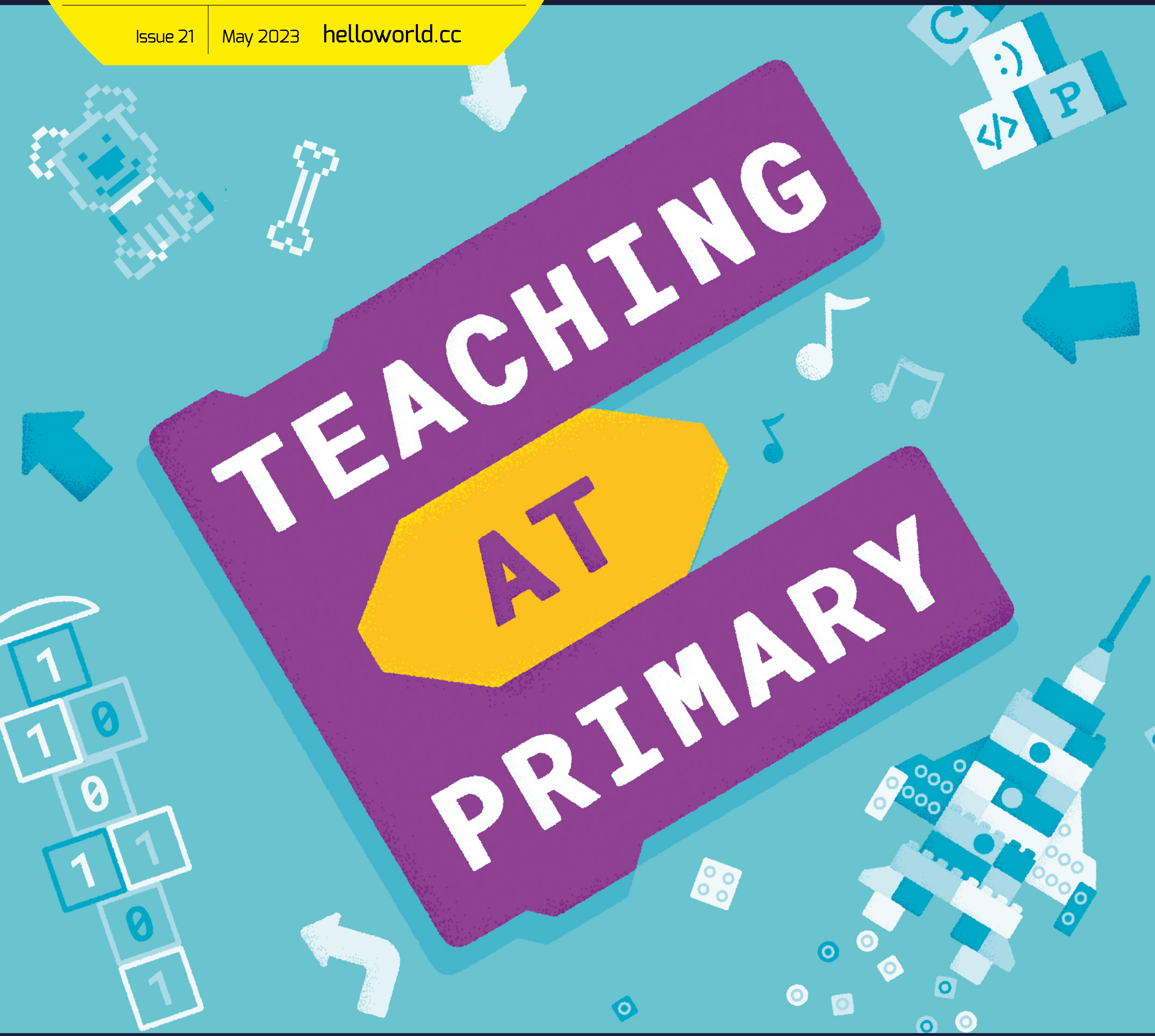
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS

Issue 21 | May 2023 helloworld.cc

TEACH FIRST
Delivering computing pedagogies

PROGRAMMING WITH DYSLEXIA
Supporting dyslexic learners to thrive

GAME JAMS
Hands-on game development



PLUS **COOLEST PROJECTS** · TYPING SKILLS · **METAPHORS AND ANALOGIES** · EXPERIENCE AI · AGILE METHODOLOGY · **ADA COMPUTER SCIENCE** · WHITEBOARDS FOR FORMATIVE ASSESSMENT · **TURTLESTITCH** · FICTION BOOKS · **ENGAGING CYBERSECURITY ACTIVITIES** · FILTER BUBBLES

Raspberry Pi Foundation

Teach and learn with the Raspberry Pi Foundation

Free for everyone anywhere in the world

Teachers' resources

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

raspberrypi.org/teach

Project library

Browse 200+ online project guides that help young people create with digital technologies.

projects.raspberrypi.org

Research to practice

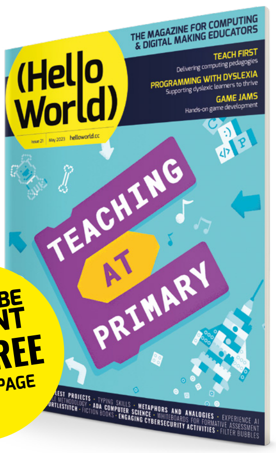
Explore our research into what works best in teaching school-age young people about computing.

raspberrypi.org/research

Coding at home

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

raspberrypi.org/learn


THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
(Hello World)
TEACH FIRST
PROGRAMMING WITH DYSLEXIA
GAME JAMS
TEACHING AT PRIMARY

SUBSCRIBE IN PRINT FOR FREE
TURN TO PAGE 46

HELLO, WORLD!

How are you preparing young children for a digital world? This is the question posed by Sway Grantham on page 26, and which this primary-teaching-themed issue responds to with inspiration and ideas. It is vital that children have a solid foundation of digital literacy skills and conceptual computing understanding and knowledge upon which to build as they grow. Technology is here to stay, and as Sethi De Clercq points out on page 31, we need to be preparing our youngest learners for jobs and circumstances that don't yet exist.

This issue traverses cross-curricular project ideas to keep young learners engaged (page 42), perfecting typing skills in the elementary classroom (page 36), and using picture books to introduce programming concepts to children (page 32). Neil Rickus and Catherine Archer offer toolkits for new and experienced computing primary teachers (pages 28 and 40), and both Chris Lovell and Peter Marshman explore approaches to improving diversity in computing (pages 37 and 44).

The focus of this issue of Hello World was born out of readers' requests for more content aimed at primary teachers. We love to hear what we can do to continue making Hello World relevant for you, so please get in touch at **contact@helloworld.cc** with your suggestions.

Finally, this is my last issue as editor of Hello World! It has been an absolute pleasure connecting with readers and working with the wonderful contributors who make this magazine so special. I look forward to seeing the next issue and cheering on from the sidelines. Bye for now!

Gemma Coleman
Editor

FEATURED THIS ISSUE

VICTORIA BERKOWITZ

Victoria is a high-school teacher in New York, USA. On pages 62–63, she outlines how she implemented an agile methodology in her robotics classes.

JARED RIGBY

Jared is a computer science teacher in Beijing, China. He shares how he ran a game jam to engage his students on pages 54–55.

MARGARET LOW

On page 41, Margaret, who works at the University of Warwick, UK, shares the value of programmatic embroidery.

(HW)

Hello World is the official magazine of the Raspberry Pi Foundation

Raspberry Pi Foundation

EDITORIAL

Editor
Gemma Coleman
Subeditors
Louise Richmond and Amy Rutter
Subscriptions
Joshua Crossman
Social Media
Neena Patel

DESIGN

criticalmedia.co.uk
Head of Design
Lee Allen
Designers
Olivia Mitchell, Sam Ribbits
Photography
The Raspberry Pi Foundation, Adobe Stock
Graphics
Rob Jervis
Cover
© Michael Parkin, Folio Art

CONTRIBUTORS

Carmella Abela, James Abela, Polly Akhurst, Catherine Archer, Rachel Arthur, Louie Barnett, Victoria Berkowitz, Simon Carter, CAS Assessment Working Group, Katharine Childs, Sethi De Clercq, Gemma Coleman, Michael Conterio, Nicki Cooper, Rachael Coultart, Josh Crossman, Mike Deutsch, Catherine Elliott, Tina Fountain, Helen Gardner, Tracy Gardner, Ben Garside, Sway Grantham, Ben Hall, Graham Hastings, Chris Lovell, Margaret Low, Duncan Maidens, Peter Marshman, Ajak Mayen Jok, Ian Needham, Kaye North, Alan O'Donohoe, Nicolai Pöhner, Neil Rickus, Jared Rigby, Chris Roffey, Peter Rutherford, Daljit Shoker, Katie Sparks, Jonathan Towers, Wariara Waireri, Jane Waite, Juliet Waters, Bobby Whyte, Phil Wickins, Preeti Williams, Tig Williams, Matthew Wimpenny-Smith

Supported by ORACLE

Contributing Partner
bcs The Chartered Institute for IT Academy of Computing
CAS

(HW) CONTENTS

14–44

COVER FEATURES

PRIMARY COMPUTING
A series of articles exploring computing for our youngest learners

NEWS, FEATURES, AND OPINION

{code club}

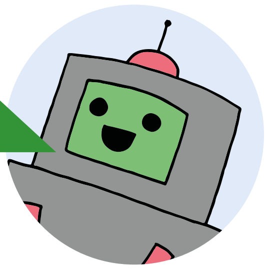
Start a Code Club in your school

Code Club provides everything you need to run coding clubs for young people aged nine and over, using a range of fun, free, and easy-to-follow project paths.

- ✓ Code Club is free and flexible
- ✓ Code Club's step-by-step Scratch, Python, and HTML & CSS projects develop independent coding skills whilst sparking imagination and creative thinking
- ✓ Our free online training and workshops are perfect for teachers who are new to coding

"Code Club provides a sense of achievement and satisfaction as children bring their coding projects to life. It's an inclusive space that fosters fun and builds friendships, whilst allowing our students to develop patience, resilience, and perseverance."

– Fiona Lindsay, Teacher



Anyone can start a Code Club – join our global community and help young people discover their world through code!

Get involved at codeclub.org

SKILLS FOR A REFUGEE COMMUNITY

Amala Education and the Raspberry Pi Foundation have launched a new vocational skills course to support displaced learners in Kenya

Polly Akhurst, Louie Barnett, Ajak Mayen Jok, Wariara Waireri, Ben Hall, & Sway Grantham

The Raspberry Pi Foundation is working in partnership with Amala Education (amalaeducation.org) to pilot a vocational skills course for displaced learners aged 16–25 in Kakuma refugee camp, Kenya. The camp was set up in 1992, following a civil war in neighbouring South Sudan. Today, 2 million people live in the camp, and 61 percent are 18 and younger.

Developing the course

Amala wanted a course that would develop its learners' digital literacy within its agency-based learning model, which teaches young people digital skills to improve their communities. Before creating the course, the Raspberry Pi Foundation held focus groups with facilitators and learners in Kakuma camp in an effort to understand their needs and help them adapt content from their existing computing curriculum (**helloworld.cc/tcc**) and pitch it at the right level. This included adaptations based on the age of the learners, English-language levels, and cultural relevance, as well as prior knowledge, classroom environment, and internet access.

The work has culminated in a 100-hour, ten-week course called *Using Online Technologies to Create Change*. Learners start by finding out how digital devices work, using the input–process–output model. Then they move on to understanding computer networks. The course includes hands-on activities related to creating media, like filming video content and making sounds to use in a podcast. There is also some light-touch web development with HTML and JavaScript. At the end of the course, learners design and deliver a presentation that reflects the work they've completed.

Throughout the course, learners use their newly gained skills and knowledge to make their own projects to create positive change. For example, one group of learners used their photography and design skills to develop ID cards to keep Amala students safe within the camp. An ID card protects learners because they can prove their identity to their community and the police.

Over 100 people within the Amala learner community expressed an interest in this 75-person course

Making the course relatable

One of the great things about this course is that the Amala course facilitators are from within the camp, so they look, speak, and sound like the learners. Having the course facilitated by fellow refugees removes the stigmatisation that the learners are vulnerable and sets the precedent that they can do anything if they put their minds to it.

While the Raspberry Pi Foundation team worked to make the course content relevant for learners, Amala facilitators further localised the content to ensure it was relatable. Localisation is also important because it helps implement one of Amala's cornerstones: decolonising the African curriculum.

Over 100 people within the Amala learner community expressed an interest in this 75-person course. The issue of digital literacy in refugee communities is clearly more urgent and pertinent than ever. In a world where a lack of access to technology and digital skills exacerbates existing inequalities (**helloworld.cc/inequality**), it is critically important for young people who are disadvantaged to access meaningful learning opportunities. As one learner put it, "Before I joined the course, I really didn't know much about how to operate technology, but through the learning and the process, now I am able to learn something that will be beneficial for me and the people in my community." Check out the Raspberry Pi Foundation's blog for more developments with this work (**raspberrypi.org/blog**). (HW)

EXPERIENCE AI: THE EXCITEMENT OF AI IN YOUR CLASSROOM

Explore free AI resources developed in partnership with educators and researchers at the Raspberry Pi Foundation and industry experts DeepMind

Ben Garside

The Raspberry Pi Foundation is excited to introduce Experience AI (experience-ai.org), a learning programme that helps educators to inspire and engage young people when teaching artificial intelligence (AI) and machine learning (ML). This new educational programme is a collaboration with leading AI company DeepMind (deepmind.com), and offers cutting-edge resources for secondary-school teachers and students.

The importance of AI and machine learning education

Artificial intelligence and machine learning applications are already changing many aspects of our lives. From search engines, social media content recommendations, self-driving cars, and facial recognition software to AI chatbots and image generation, these technologies are increasingly common in our everyday lives.

Young people who understand how AI works will be better equipped to engage with the changes AI applications bring to the world, make informed decisions about using and creating AI applications, and choose what role AI should play in their futures (**helloworld.cc/ofsted2022**). They will also gain critical-thinking skills and awareness of how they might use AI to come up with new, creative solutions to problems they care about.

The AI applications people are building today are predicted to affect many career paths. In 2020, the World Economic Forum estimated that AI would replace some 85 million jobs by 2025 and create 97 million new ones (**helloworld.cc/WEF2020**). Many of these future jobs will require some knowledge of AI and ML, so it's important that young people develop a strong understanding from an early age.

Develop a strong understanding of the concepts of AI and machine learning with your learners

Experience AI lessons

Something we get asked a lot at the Raspberry Pi Foundation is, 'How do I teach AI and machine learning to my class?' To answer this question, we have developed a set of free lessons as part of the Experience AI programme that give you everything you need, including lesson plans, slide decks, worksheets, and videos. The lessons are aimed at secondary-school students (aged 11 to 14) and are suitable for both classrooms and settings such as coding clubs if you're an educator or volunteer outside of a school environment.

The lessons focus on relatable applications of AI and are carefully designed so that teachers of a wide range of subjects can use them. Here is a preview of what's on offer in each lesson:

1. **What is AI?:** learners explore the current context of AI and how it is used in the world around them. Looking at the differences between rule-based and data-driven approaches to programming, they consider the benefits and challenges that AI could bring to society.
2. **How computers learn:** learners focus on the role of data-driven models in AI systems. They are introduced to machine learning and find out about three common approaches to creating ML models. Finally, the learners explore classification, which is a specific application of ML.
3. **Bias in, bias out:** learners create their own machine learning models to classify images of apples and tomatoes. They discover that a limited data set will likely lead to a flawed ML model. Then they explore how bias can appear in a data set, resulting in biased predictions produced by an ML model.
4. **Decision trees:** learners take their first in-depth look at a specific type of machine learning model: decision trees. They see how different training data sets create different ML models, experiencing first-hand what the term 'data-driven' means.
5. **Solving problems with ML models:** learners are introduced to the AI project life cycle and use it to create a machine learning model. They apply a human-focused approach to working on their project, train an ML model, and finally test their model to find out its accuracy.
6. **Model cards and careers:** learners finish the AI project life cycle by creating a model card to explain their machine learning model (**helloworld.cc/modelcards**). To finish off the unit, they explore a range of AI-related careers, hear from people working in AI research at DeepMind, and explore how they might apply AI and ML to their interests.

Ben Garside, one of the Raspberry Pi Foundation's lead educators working on Experience AI, takes a group of students through one of the new lessons

> AI AND MACHINE LEARNING ARE ALREADY CHANGING MANY ASPECTS OF OUR LIVES

We've designed these Experience AI lessons with teacher support in mind, and so that you can deliver them to your learners no matter your subject area. Each lesson plan includes a section that explains new concepts, and the slide decks feature embedded videos in which DeepMind's AI researchers describe and bring these concepts to life for your learners. You can find out more about how we used research to shape the lessons (**helloworld.cc/waite2023**) and how we aim to avoid misconceptions about AI (**helloworld.cc/garside2023**).

Next steps

We're inviting teachers in the UK to participate in research to help us develop these resources further. All you need to do is sign up through our website (**experience-ai.org**), download the lessons, use them in your classroom, and give us your valuable feedback. We are really looking forward to working with you to shape the future of AI and machine learning education.

We will also offer a range of new teacher training opportunities later this year, including a free online CPD course — *Introduction to AI and Machine Learning* — and a series of AI-themed webinars. Watch this space! (HW)

INCREASING GIRLS' PARTICIPATION IN COMPUTING

Katharine Childs shares what the findings of the Gender Balance in Computing research programme mean for computing teachers

Katharine Childs

Computing teachers can access various initiatives and programmes to address the gender imbalance in computing. Nonetheless, girls are still underrepresented in formal computer science qualifications in secondary education in England. The Gender Balance in Computing (GBIC) research programme (helloworld.cc/GBICresearch), led by the Raspberry Pi Foundation, sought to determine whether any single intervention could significantly boost the proportion of girls selecting computer science. It did this by exploring the success of interventions around teaching approaches used in computing: fostering a sense of belonging; making computing relevant to real-world problems; establishing connections between formal and non-formal computing; and exploring how computer science is presented as a GCSE option (formal exams in England for pupils aged 15 to 16).

The interventions were evaluated using randomised controlled trials (RCTs). Most of the RCTs showed a positive impact that fell just short of statistical significance. However, feedback from the teachers indicated that they were enthusiastic about the methods for addressing the known barriers to gender balance introduced by the research programme, and that pupils also reacted positively.

Primary-to-secondary transition

The project collected data from over 7000 female pupils in primary and secondary schools in England. At the end of each intervention, each pupil was asked whether she intended to study computing in the future, and there was a noticeable difference in numbers between secondary girls (7 percent), and primary girls (55–57 percent) (**helloworld.cc/GBIC**). This suggests that secondary-school computing teachers need to target their efforts to engage girls in computing right from the first year of secondary school, and continue initiatives through to the point when young people decide which subjects to choose as electives. Primary teachers also have an important role to play by making sure that girls' engagement with computing is maintained all the way through to the end of primary school.

The way forward

The research results have given us at the Raspberry Pi Foundation lots of data that will help us to refine the studies and potentially combine initiatives in future studies. The lead researcher for this programme, Dr Sue Sentance, recommends that schools take a multifaceted approach to gender balance by implementing a range of the trialled approaches throughout primary and secondary education:

"This research has shown us that multiple initiatives are needed to increase the gender balance in non-mandatory computing courses. We heard from teachers participating in the study about the ways in which pupils benefitted from more collaborative teaching approaches and computing being made more relevant to the world around them. We would encourage teachers to incorporate all of the approaches used in the GBIC trials. Meanwhile, more longitudinal research is needed around the impact of combining initiatives." (HW)

FIND OUT MORE

The Gender Balance in Computing research programme ran from 2019 to 2022 and was the largest research programme of its kind to date. A team of partners from the Raspberry Pi Foundation, the Behavioural Insights Team, Apps for Good, and WISE designed, implemented, and analysed the results from the interventions, which were delivered in over 500 primary and 250 secondary schools in England.

The full reports from each intervention, as well as the teaching materials used in the interventions, have been published at **helloworld.cc/GBICresearch**.

INTRODUCING ADA COMPUTER SCIENCE

A new platform from the Raspberry Pi Foundation for learning about computer science

Duncan Maidens

The Raspberry Pi Foundation is excited to launch Ada Computer Science, the new online learning platform for teachers, students, and anyone interested in learning about computer science (adacomputerscience.org). With the rapid advances being made in AI systems and chatbots built on large language models, such as ChatGPT (helloworld.cc/chatGPT), it's more important than ever that all young people understand the fundamentals of computer science. The Raspberry Pi Foundation aims to enable young people all over the world to learn about computer science by providing access to free, high-quality, and engaging resources that both students and teachers can use.

A tool for classwork and homework

A partnership between the Raspberry Pi Foundation and the University of Cambridge, Ada Computer Science offers comprehensive resources covering everything from algorithms and data structures to computational thinking and cybersecurity. It also has nearly 1000 rigorously researched and automatically marked interactive questions to test students' understanding. Ada Computer Science is improving all the time, with new content developed in response to the latest research and user feedback. Whatever your interest in computer science, Ada is the place for you.

If you're teaching or studying for a computer science qualification at school, you can use Ada Computer Science for classwork, homework, and revision. Computer science teachers can select questions to set as assignments for their students and have them marked directly. The assignment results help both teachers and students understand how well they have grasped the key concepts and identify areas where they would benefit from further tuition. Students can learn with the help of written materials, concept illustrations, and videos, and they can test their knowledge and prepare for exams.

Mapping Ada's topics

Ada Computer Science builds on work the Raspberry Pi Foundation has done to support the English school system as part of the National Centre for Computing Education, funded by England's Department for Education. The topics on the website map to exam board specifications for England's computer science GCSE and A level, and will map to other curricula in the future.

In addition, we want to make it easy for educators and learners across the globe to use Ada Computer Science. That's why each topic aligns with the Raspberry Pi Foundation's own comprehensive taxonomy of computing content for education (**helloworld.cc/RPFtaxonomy**). This is independent of the English curriculum, and organises the content into eleven strands, including programming, computing systems, data and information, artificial intelligence, creating media, and societal impacts of digital technology (you can explore the taxonomy more in Hello World's *The Big Book of Computing Content*, **helloworld.cc/bigbook2**). If you are interested in how we can adapt Ada Computer Science specifically for your region, exam specification, or specialist area, please contact us at **hello@adacomputerscience.org**.

Ada Computer Science takes its name from Ada Lovelace, of course, who was credited as being the first computer programmer. We hope that these resources will support students to become confident and engaged computer scientists, just like their namesake. (HW)

WHY USE ADA COMPUTER SCIENCE AT SCHOOL?

Ada Computer Science enables teachers to:

- Plan lessons around high-quality content
- Set self-marking homework questions
- Pinpoint areas to work on with students
- Manage students' progress in a personal markbook

Students get:

- Free computer science resources written by specialist teachers
- A huge bank of interactive questions designed to support learning
- A powerful revision tool for exams

In addition:

- The topics include real code examples in Python, Java, VB, and C#
- The live code editor features interactive coding tasks in Python
- Quizzes make it quick and easy to set work

© Julien Eichinger/stock.adobe.com

MORE NEWS FROM HELLO WORLD

Introducing the Hello World newsletter, a monthly round-up of news from the Raspberry Pi Foundation and Hello World magazine

Gemma Coleman

Since launching Hello World magazine six years ago, we've heard from many readers that they'd like to see more great content for computing educators while they wait for the next issue to be published. We've answered this call with an exciting podcast in which we chat with educators around the globe about all things computing and digital making (helloworld.cc/podcast). We've also introduced a blog on the Hello World home page in which we share some of our favourite past articles (helloworld.cc).

A monthly newsletter

Now, Hello World is launching a monthly newsletter with your fix of computing education news from the Raspberry Pi Foundation and Hello World (**helloworld.cc/HWnewsletter**). As the official magazine of the Raspberry Pi Foundation, we want to share news with our readers and the wider community about all the great research, resources, community stories, and events from the Foundation, as well as updates and interesting articles from Hello World.

Every month, we'll be curating a number of these stories, with a bite-sized summary of each initiative and relevant links so you can explore more in your own time.

Keep up with all the education news from the Raspberry Pi Foundation and Hello World by signing up at **helloworld.cc/HWnewsletter**. If you already subscribe to the Pi Learn newsletter, you don't need to do anything; this newsletter will replace Pi Learn and you will be automatically subscribed. We hope you enjoy our first newsletter, which is out in June! (HW)

" KEEP UP TO DATE WITH THE FOUNDATION'S RESEARCH, RESOURCES, STORIES, AND EVENTS

MOONHACK: A CHALLENGE THAT'S OUT OF THIS WORLD

The 2023 Moonhack Coding Challenge will highlight the innovations of young coders from around the world on a theme of Space and Innovation

Kaye North

Here's what one Code Club leader had to say after their group took part in last year's Moonhack event (moonhack.com): "The opportunity for students to explore a range of sustainability topics and gain new knowledge and understanding while building, utilising, and enhancing their coding skills through the Scratch platform is ideal!" Moonhack is a free international event, run by Code Club Australia (codeclubau.org), that brings kids aged 8–15 together for two weeks of coding fun. Over the last eight years, over 190,000 children have coded Moonhack projects, with each annual challenge presenting a unique theme about the world or space. In 2022, a record 44,000 kids from 64 countries rose to the challenge and completed at least one of six coding projects created especially for the challenge. Moonhack is set to return in October 2023 — this time, to inspire kids to explore innovations in space.

Digital solutions in space

Moonhack provides an opportunity for kids in clubs, classrooms, and at home to engage with developing digital solutions to everyday problems in a unique way. Last year, for example, the challenges focused on satellites and their place in our world. Children explored the UN's Sustainable Development Goals and looked at how satellites are being used to improve our way of life. Six different projects, focused on building computational thinking and coding skills, were produced to inspire kids to create and innovate (**helloworld.cc/moonhackprojects**). These spanned from creating a clicker game, an animation, and a puzzle; to retelling an indigenous dreamtime story; to creating a forest alarm using micro:bits; and finally, to using Python to determine the exact times of twilights.

The challenge is for all children, from first-time coders to coding whizz-kids. The projects are designed so that learners can use them to gain new coding skills, or they can be built upon by an individual with their own ideas, creativity, and coding knowledge, and their unique views on the topic. To support educators, each project is explored in detail in a blog post. This outlines the importance of the project and why it is featured in the challenge. It also provides background information, links to further knowledge and resources, and ideas on how to extend the project.

Blasting off for 2023

This year's global challenge will run for two weeks in October. There will be seven brand new projects themed around Space and Innovation, with kids using Scratch, Python, and micro:bit to develop and showcase space inventions. A new addition for 2023 will be a project presented as a design brief. The project will highlight a current problem in space (such as space junk), and young coders will design their own coded solution. This could be creating a game or animation that informs the audience about the problem, or using Python or Scratch to demonstrate a solution that they invent.

As a global community, we want children to share their coded solutions and ideas with one another, so projects with sharing permission will be hosted in our Moonhack Studio in Scratch (**helloworld.cc/moonhackstudio**) and on our website. This year we are also introducing Moonhack code-along videos, Moonhack meet-ups, and other ways to collaborate online.

Code Club exists to democratise technology education: to reach kids and adults who may not have access to digital education resources; to grow the representation of minority groups in technology; and to tell stories that fuel these objectives. Moonhack is just one of our initiatives that's helping us to achieve these goals, and we hope you'll join us for an out-of-this-world experience.

*To keep up to date with new developments, the official announcement of dates and themes, and when registration opens, subscribe to the Moonhack newsletter (**helloworld.cc/moonhacknews**) and bookmark the Moonhack website (**moonhack.com**).* (HW)

The Tagai State College Moonhack team

COLUMN

AN INCLUSIVE COMPUTING CURRICULUM FOR YOUNGER LEARNERS

Catherine Elliott explores the key strategies and components of an inclusive curriculum educators can use to support our younger learners

When designing an inclusive curriculum for computing, we should ask ourselves, 'Which pupils are finding the subject most difficult?' and 'Why are they finding it hard?' The answers will show us how to proceed. Do children struggle with the fine motor skills needed to use a mouse? Do they have great practical skills, but find it hard to remember key vocabulary? Do they carry any fundamental misconceptions with them into their learning? In this article, I examine some of the key components of an inclusive computing curriculum for children aged four to eleven, and share strategies for ensuring all pupils are included in the subject.

Key skills

The skills required to access a computer are essential if pupils are to achieve in the rest of the curriculum. If a child cannot log on or use a mouse or keyboard effectively, they cannot design a poster or write a program in a desktop application. It is therefore beneficial to spend some time developing these key skills with younger pupils and targeted groups of learners, to build up fluency. This could involve taking smaller groups to practise logging in, using creative tools such as a paint application to practise mouse skills, or using unplugged activities to focus on a skill before using computers. For example, you could print out pictures of keyboards for pupils to practise finding individual letters or spelling out high-frequency words in the classroom. Do pupils have opportunities to practise their key skills in other areas of the curriculum too?

Some pupils may benefit from assistive technology, such as a rollerball mouse or BigKeys keyboards (**bigkeys.com**). For others, it could be useful to have a laminated set of instructions with supporting images for common processes (such as logging in, opening and saving work, etc.) next to the computer. Are there also any school-wide barriers that you can help them overcome? For example, do five-year-olds need to type in a long username and complex password each time they use a computer? How many steps does it take to save work in their own folder?

Catherine Elliott is the SEND lead for the Sheffield eLearning Service (**sheffieldclc.net**), and since the announcement of the computing curriculum in 2013, she has been working on ways to make the subject accessible for 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**).

The language of computing

Over the last few decades, the idea has gained traction that every teacher is a teacher of language, and this is no less true in computing than in other less practical subjects. Have you identified the key learning and language for each unit you're teaching, which all pupils should know?

Many pupils will need extra support with remembering key terms and comprehending the meaning of abstract concepts. The strategies to do this are the same as in other subjects: preteaching key vocabulary, using image-supported glossaries, introducing terms slowly, and creating plenty of opportunities to recall them during lessons. It is also important that teachers start to model the key language during early years education. This isn't always easy for non-specialists to do, so computing leads also need to spend time with staff ensuring that they can identify the key terms in each topic and what they mean at an age-appropriate level.

In the last issue of Hello World, Alex Hadwen-Bennett wrote about the role of physical gestures made by visually impaired learners when describing the flow of a program they had written in a tactile programming language (page 76, **helloworld.cc/20**). Other research indicates that as well as gestures made by teachers helping students to remember key learning, students who gesture themselves when explaining something to a partner have better retrieval (**helloworld.cc/rogers2020**).

Teachers can help guide pupils by providing physical gestures to represent key concepts — for example, a looping gesture to represent repetition and loops, or a branching gesture to indicate selection. Educators can also encourage learners to physically trace the flow of programs to help with comprehension, whether it be pointing to the simple linear steps of a Bee-Bot program,

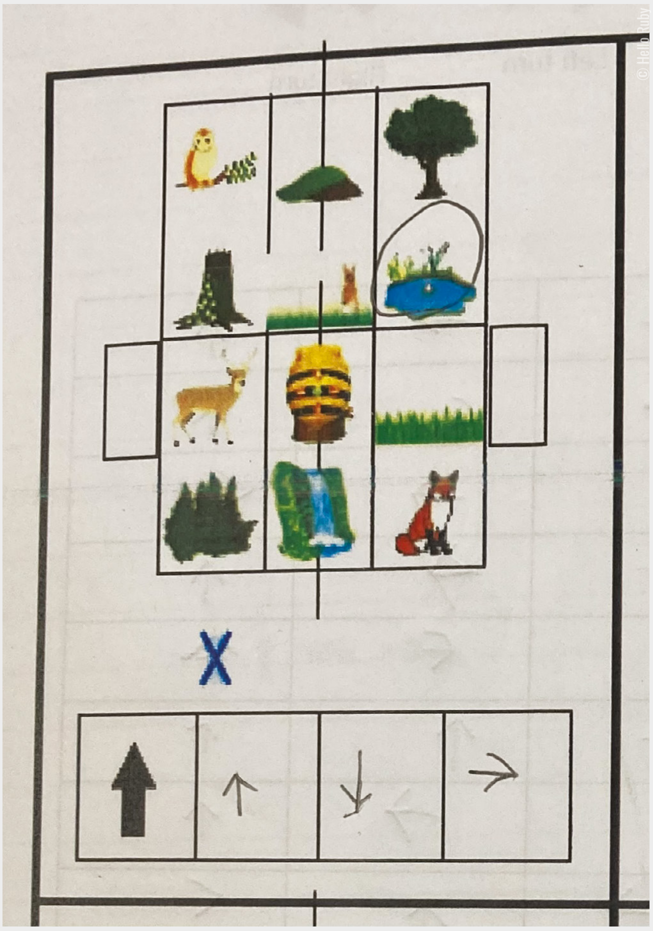
> PROVIDE A SAFE SPACE TO MAKE MISTAKES

or the more complex flow in Scratch. I will always remember seeing students sitting formal exams recreating the gestures of a preposition song in German to help them remember the terms. Can you create similar song-based mnemonics with accompanying gestures to help pupils remember something like input and output devices?

Anticipating misconceptions

Understanding which parts of the curriculum pupils are likely to find difficult helps with anticipating misconceptions and other barriers to learning. You can then better support pupils to avoid the pitfalls. A couple of ways of doing this are by providing a safe space for pupils to make mistakes and modelling how to correct them, and explicitly signposting from the start possible difficulties and how to avoid them.

For example, from experience, I know that when I am teaching stop-motion animation, children massively underestimate the number of frames they need to take in order to create even the shortest film, and overestimate how far to move their characters between frames. I can model these things to the class very carefully, but often they have to experience it for themselves. If they are given time to tinker with a Lego figure or piece of plasticine at the start of the project, pupils can create an animation without the pressure of it being the final piece. You can then discuss as a class how to improve individual animations before children go off and put their learning into practice. For some pupils, you could have an image or audio-supported guide available to show how far to move characters, and provide a specific number of frames to work towards.



Identifying any challenging curriculum areas helps with anticipating misconceptions

Effective formative assessment is necessary to identify misconceptions and difficulties. It isn't always easy to spot these in group work, but by providing unplugged challenges, such as predicting the outcome of code or a Parson's Problem, we can quickly see what has and hasn't been understood. For example, in the image of the Bee-Bot challenges above, I can quickly identify that this pupil doesn't understand what happens when the 'right' button is pressed on the Bee-Bot (the robot would rotate 90 degrees on the spot, not move across to the right to the pond). More modelling and explicit practice around turns is therefore required.

These are only a few examples of how to create a more inclusive curriculum for young learners with special educational needs and disabilities. You should also consider careful chunking and scaffolding of activities, and provide a variety of ways to access learning, as described by the Universal Design for Learning framework, which I have written about in previous issues. Above all, we should be encouraging every learner to discover their joy for the subject by allowing them to show off their strengths and succeed in computing. (HW)

PROGRAMMING MULTIMODAL STORIES

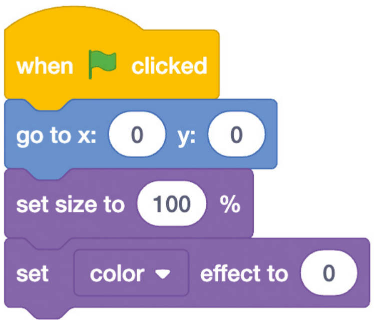
STORY BY Bobby Whyte

Cross-curricular projects are common in computing education. As well as demonstrating the applications of programming in different subject areas (such as science or mathematics), they can also support learners to develop their problem-solving processes. For instance, teachers might use ScratchMaths (helloworld.cc/scratchmaths) to explore mathematical ideas with learners. This approach isn't entirely new; in fact, Seymour Papert anticipated that Logo, a precursor to modern block-based programming languages, would enable young people to learn about mathematics in an intuitive environment, in much the same way as a young person learns their native tongue in their home country. He called this environment 'Mathland'. What other lands might we consider exploring in computing?

In my PhD research project, I investigated the land of literacy — more specifically, stories — to consider how programming and storytelling could be meaningfully integrated for young learners aged seven to eleven. Using Scratch, we conducted research in schools by testing a curriculum unit based on multimodal stories.

USEFUL RESOURCES

- Creative Computing Curriculum from the Harvard Graduate School of Education: **helloworld.cc/CCC**
- Scratch Encore from UChicago STEM Education: **helloworld.cc/scratchencore**



```
when [flag] clicked
go to x: (0) y: (0)
set size to (100) %
set [color v] effect to (0)
```

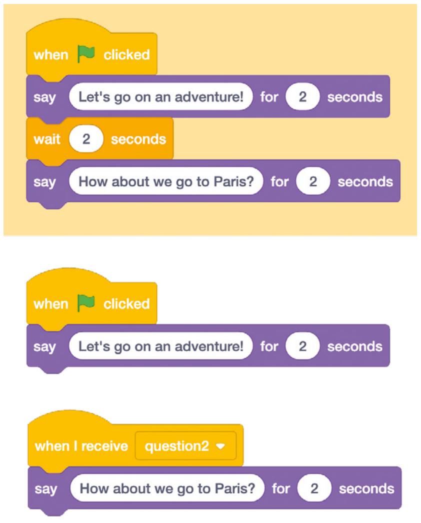
Figure 1 Resetting sprite states is important for repeat execution

Programming multimodal stories

A multimodal story is a little different from a typical story learners might write. Where stories written in class use one (monomodal) mode of communication, such as text, a multimodal story allows learners to use multiple modes to communicate meaning. For instance, learners might use images, sounds, and videos. In Scratch, some of the available modes lend themselves well to telling stories: backdrops can be used to represent settings, sprite costumes to represent different character states, and audio files to represent sound effects.

Programming multimodal stories requires learners to consider both what story they want to tell, and how they are going to tell it. In my research project, I aimed to support learners in applying a range of programming features to support their storytelling practices. Here are some strategies that were used to help promote different programming strategies as learners considered which stories to tell and how to tell them:

- **Program execution:** multimodal stories should look the same each time they are run, or they might lose coherence. You can demonstrate the importance of resetting a program state by discussing how the end user (or reader) might encounter learners' projects. Each time an end user runs a program, any program states, such as a character's x- and y-coordinates, a sprite's show/hide state, or any graphical effects, should be automatically reset (see **Figure 1**), or the reader might not follow the story's events.

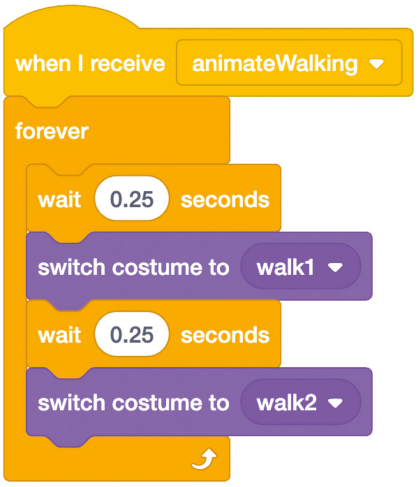


```
when [flag] clicked
say [Let's go on an adventure!] for (2) seconds
wait (2) seconds
say [How about we go to Paris?] for (2) seconds
```

```
when [flag] clicked
say [Let's go on an adventure!] for (2) seconds
```

```
when I receive [question2 v]
say [How about we go to Paris?] for (2) seconds
```

Figure 2 Two approaches for creating dialogue in Scratch



```
when I receive [animateWalking ▾]
forever
    wait (0.25) seconds
    switch costume to [walk1 ▾]
    wait (0.25) seconds
    switch costume to [walk2 ▾]
```

Figure 3 Animations are a practical way of teaching about loops

- **Event-based programming:** to tell multimodal stories correctly, learners need to sequence each element of their story in a particular order. This provides an opportunity to demonstrate different strategies for coordinating sprites, backdrops, and any other story features. For less experienced learners, this might include using wait blocks to coordinate elements in sequence, such as two characters having a conversation. For more experienced learners, this could include broadcast blocks to coordinate multiple elements in parallel, such as sprites talking while their costumes loop to simulate movement (see **Figure 2**).

- **Loop functions:** encouraging learners to animate characters or objects provides an opportunity to explore how different loops can be used. For instance, an infinite loop cycling through two costumes could simulate a character walking (see **Figure 3**), while using a 'repeat until' block could toggle multiple loops for different character states, such as a character breathing and then walking.

- **Story selection:** remember choose-your-own-adventure stories? Using a familiar story (such as a hero's adventure), learners could program alternative endings as a way to encounter the concept of selection. Using 'if' statements to control a branching narrative with multiple endings, learners can enable end users to determine the outcome of their story (see **Figure 4**).

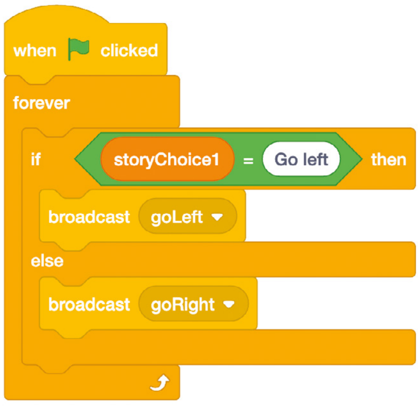
" HOW CAN PROGRAMMING AND STORYTELLING BE MEANINGFULLY INTEGRATED FOR YOUNG LEARNERS?

- **Variable mad libs:** mad lib stories are interactive stories into which a reader inserts words of their choice to create their own unique version. Variables could act as placeholders for key elements in a story, such as the main character's name, setting, or adjectives used in speech; learners could program a story in which the end user can interactively tell a unique story each time. This also provides an opportunity for learners to work with operators within text (see **Figure 5**). They could also use variables to set key parameters in a text, such as the setting or outcome.

In my research, I found that multimodal stories are a useful approach for teaching learners about different concepts and practices in programming and storytelling. It is also encouraging to see lots of new resources to teach programming through storytelling approaches; one notable example is some experimental blocks that have recently been made available from Scratch Lab (**helloworld.cc/scratchlab**) to add animated text to projects. How could you use multimodal stories to support your computing lessons? (HW)

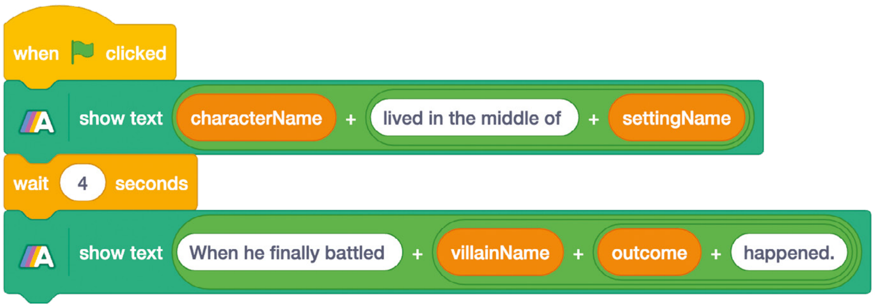
FURTHER READING

- ✓ Whyte, R., Ainsworth, S., & Medwell, J. (2020). Designing Multimodal Composition Activities for Integrated K-5 Programming and Storytelling. In M. Gresalfi & I. S. Horn (Eds.), *The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences* (pp. 1317–1324). International Society of the Learning Sciences. **helloworld.cc/issue21insights1**
- ✓ Whyte, R., Ainsworth, S., & Medwell, J. (2019). Designing for Integrated K-5 Computing and Literacy through Story-making Activities. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (ICER 2019) (pp. 167–175). **helloworld.cc/issue21insights2**



```
when ⚑ clicked
forever
    if <(storyChoice1) = (Go left)> then
        broadcast [goLeft ▾]
    else
        broadcast [goRight ▾]
```

Figure 4 Choose-your-own-adventure stories are an accessible way to teach learners about selection



```
when ⚑ clicked
show text (characterName) + (lived in the middle of) + (settingName)
wait (4) seconds
show text (When he finally battled) + (villainName) + (outcome) + (happened.)
```

Figure 5 Interactive mad lib stories can be created using variables as story elements

TIPP&SEE FOR EQUITY

STORY BY Mike Deutsch

Computing education programmes continue to grow across the world, and more primary students gain access to coding lessons each year. Once in place, though, we know that such lessons are not uniformly effective. Students with economic and academic disadvantages often learn less from their programming lessons than their more privileged counterparts. This is unjust; we cannot be satisfied by carrying forward the inequities our students already face. As Dr Jean Salac, a researcher addressing this gap, declared in a recent research seminar: "Access isn't enough. We need equitable learning outcomes" (helloworld.cc/salacseminar).

Equity through pedagogy

One way to achieve this is through pedagogy — our choice of how we teach. In this light, Salac and her colleagues have created TIPP&SEE (Title–Instructions–Purpose–Play–Sprites–Events–Explore), a teaching strategy inspired by techniques used in primary-reading comprehension (**helloworld.cc/franklin2020**).

Designed for teaching programming in Scratch, TIPP&SEE gives new programmers additional scaffolding in the first two phases of a Use–Modify–Create sequence. Learners begin with an already-built program, reading its Title page and Instructions. They consider the Purpose of the program, and they Play it and carefully observe what it does. They then work their way into the code, paying specific attention to the Sprites and the Events that make things happen. Finally, they Explore the code more actively, modifying certain blocks and noting how this changes the program's behaviour.

The researchers have found that this extra metacognitive structure, which they call deliberate tinkering, better prepares students for mastery. In their studies, TIPP&SEE dramatically improved students' ability to comprehend programs, interpret them, and predict their behaviour. Students using TIPP&SEE also showed improved skill at open-ended creation, writing longer block sequences and using a greater variety of blocks than students without the scaffolding.

While TIPP&SEE had these effects in all the groups studied, it had the most significant effect on students with disadvantages that were completely unrelated to programming. Here is its value with respect to equity. Using the TIPP&SEE strategy narrowed the learning gaps that were otherwise experienced by students with economic disadvantages, low proficiency in reading and mathematics, and special needs and disabilities.

TIPP&SEE
Title
Instructions
Purpose
Play
Sprites
Events
Explore

TIPP&SEE is a teaching strategy inspired by techniques used in primary-reading comprehension

Instructional leverage

The researchers themselves have not identified *how* the TIPP&SEE strategy achieves what it does, but they suspect it lies in the way it makes implicit learning strategies explicit. Here are some aspects of TIPP&SEE that give it leverage for new programmers:

- **It strikes a pedagogical balance:** TIPP&SEE occupies a middle ground between the creative constructionist approach and highly structured direct instruction. It opens access for learners who do not thrive in either extreme.
- **It brings literacy to programming:** TIPP&SEE applies the reading strategies of previewing and text structure to the peculiar world of code. It helps learners arrive in a new program and orient themselves, then focus on key aspects of Scratch that they might otherwise lose in the clutter.
- **It emphasises metacognition:** TIPP&SEE guides learners to monitor and regulate their own thinking and progress, as expert learners do.

FURTHER READING

- ✓ Franklin, D., Salac, J., Crenshaw, Z., Turimella, S., Klain, Z., Anaya, M., & Thomas, C. (2020). Exploring student behavior using the TIPP&SEE learning strategy. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (pp. 91–101). **helloworld.cc/issue21insights3**

THE STRATEGY WAS FOUND TO REDUCE LEARNING GAPS

We want all our students to emerge from their introductory programming education with both conceptual understanding and creative skill. It is a matter of equity that we find strategies to do this, especially where our students face disadvantages. TIPP&SEE is one such strategy that is based on trusted methods and shows concrete results. You can find a catalogue of Scratch activities that use the TIPP&SEE teaching strategy at **helloworld.cc/TIPPSEEscratch**. (HW)

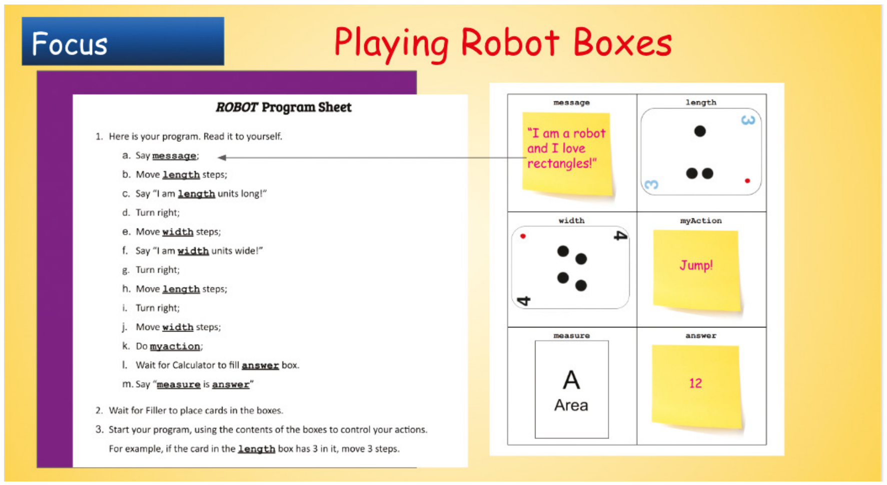
A MAP FOR MASTERING VARIABLES

STORY BY Juliet Waters

Teaching variables is always a challenge. Even if students understand how variables work in a mathematical context, understanding how they work in programming often requires new ways of thinking. Having a strategy to make visible the invisible stumbling blocks of unfamiliar thinking is a first step in supporting learners.

One strategy is the learning trajectory identified by Carla Strickland and Katie Rich as part of their work in mapping paths for all the foundational concepts in programming (**helloworld.cc/trajectories**). A learning trajectory describes the expected progression of learning through a particular concept, such as sequences or variables. It can help teachers choose the appropriate materials to balance conceptualisation challenges with hands-on practices. Strickland and Rich identified four levels of thinking that learners must navigate in order to understand variables: data user, data storer, variable user, and variable creator (**helloworld.cc/rich2020**).

At the starting gate is the data user who can represent information with mental objects, and act on them. Think of the school-age child who has newly mastered counting. They may be able to repeat numbers from memory, and perhaps even in the right order. But they are not using the numbers as objects until they understand that there are quantities associated with them. Once they can make the association, they become data users, the first level of thinking, able to act on these numbers as mental objects in basic arithmetic.



Focus

Playing Robot Boxes

***ROBOT* Program Sheet**

1. Here is your program. Read it to yourself.
   a. Say **message**;
   b. Move **length** steps;
   c. Say "I am **length** units long!"
   d. Turn right;
   e. Move **width** steps;
   f. Say "I am **width** units wide!"
   g. Turn right;
   h. Move **length** steps;
   i. Turn right;
   j. Move **width** steps;
   k. Do **myaction**;
   l. Wait for Calculator to fill **answer** box.
   m. Say "**measure** is **answer**"
2. Wait for Filler to place cards in the boxes.
3. Start your program, using the contents of the boxes to control your actions. For example, if the card in the **length** box has 3 in it, move 3 steps.

message	length
"I am a robot and I love rectangles!"	3
width	**myAction**
4	Jump!
measure	**answer**
A Area	12

New levels of thinking

Learners who can then mentally place the objects representing data into containers represented by the variables are activating the data storer level of thinking. As part of their Everyday Computing programme, Strickland and Rich have developed Robot Boxes (**helloworld.cc/robotboxes**), an unplugged activity to identify different types of variables in an unplugged sequence, placing them in boxes, and asking students to imagine how changing the variables would impact the output (see image).

The crucial difference between a data storer and a variable user is the change in perception that learners experience when they move from acting *like* a computer that stores data, to acting *on* the computer so that the computer is now doing the work of updating the variable. Here, Strickland and Rich developed a series of Scratch projects called Action Fractions (**helloworld.cc/actionfractions**). The project helps primary students understand how to add simple fractions with the same denominator. In examining how an efficient sequence of Scratch code updates the denominator with a user-input prompt, students also learn the advantages of understanding how variables work. The computer does that work for them! Learners who can conceptualise a program that updates a variable in a general sense, rather than in a specific instance, are then on their way to the finish line, thinking like a variable creator. At this level, a learner will be able to take on more highly specified cognitive actions. They will be able to understand how to initialise (set up a variable with the right data and the right type of data, so that the program is ready to run), how to update a variable, and how to turn those actions into programming commands.

With that milestone achieved, students are now ready to set out on a new voyage, beyond thinking like a programmer, towards all the possibilities available to them as they become a programmer. (HW)

FURTHER READING

✓ Rich, K. M., Franklin, D., Strickland, C., Isaacs, A., & Eatinger, D. (2020). A Learning Trajectory for Variables Based in Computational Thinking Literature: Using Levels of Thinking to Develop Instruction. *Computer Science Education, 32*(2), 213–234. (**helloworld.cc/issue21insights4**)

In each issue, the CAS Assessment Working Group share their current activities, findings, or reports relating to computing assessment.

In this column, they share one member's experience of using the 'levels of abstraction' approach to programming as part of assessment.

ALGORITHMS IN ASSESSMENT

The CAS Assessment Working Group discuss the power of using levels of abstraction as part of programming assessment

WRITTEN BY Phil Wickins

You may have heard about design algorithms and their importance within primary computing in general. Jane Waite and others have written about this extensively — please read their research on using the levels of abstraction (LOA) approach in programming (helloworld.cc/LOA), as it is fascinating and helpful. This article is more anecdotal, and tells of how encouraging pupils to create an algorithm before turning on the computer at the start of a project can play an extremely useful part in your assessment of their skills.

Let me tell my story of how using LOA changed the way I deliver computing in the classroom. My understanding is that this approach mirrors the stages of a design and technology project: task, plan, build, and evaluate. In computing, this can translate to task, design, code, and run. The design stage in this process is an opportunity for pupils to flex their computational thinking muscles, to plan how they will solve their problem or achieve their goal, before going ahead and using the power of computers to do it. In other words, in the design stage, they create an algorithm, which they will then implement as code on a computer later.

Prior to using the LOA model, I would simply inform pupils of the task and then go straight to the coding stage: "This half term we will be creating maze games in Scratch. Let's get the laptops out and begin!" Then I'd either model the code on my screen for everyone, or let my higher-ability students get on with it while I supported others in small groups. What I had inadvertently achieved there was promoting 'tinkering', or trial and error, to pupils who were brave enough to give it a go, while for the rest, I was essentially doing the work for them. Where, in this approach, were pupils using computational thinking skills to solve the problem first, before implementing it on their devices? It's a bit like writing stories in literacy; we wouldn't hand out the books and ask pupils to begin writing a complete story off the top of their heads. We'd discuss setting and character descriptions, prepare word banks, and write a plan (everyone loves a good old story mountain, right?). I was missing out the story mountain and going straight to the first draft!

Getting started with design algorithms

So, what does a design algorithm look like? Well, it is anything that a pupil can use to describe in detail what they are going to do and how they are going to do it. This could be step-by-step instructions of how their project will run, possibly the order in which they will code it, how it will look, how the user will interact with it (input/output), what kind of variables it will have, and so on. I like to give my younger programmers a template with boxes in which to draw what their Scratch game might look like; prompts to remind them to state which keys will move their sprites around the screen; and what the aim of the game is (**Figure 1** shows a template for a storyboard algorithm). Once students get to upper-primary level (aged seven to eleven) and are familiar with the process, I simply hand out blank sheets of paper and give them the freedom to design the algorithm however they want.

There are multiple reasons why using a design algorithm benefits the programmer: it

ABOUT THE CAS ASSESSMENT WORKING GROUP

The Computing at School (CAS) Assessment Working Group is a CAS community aiming to provide clear statements of informed opinions relating to current issues of formative and summative assessment in the teaching of computing, reflecting academic and professional understanding. The group presents the teacher's voice but is informed by colleagues in awarding organisations, commerce, and academia (**helloworld.cc/casawg**).

holds them to account; it gives them a chance to remember where they got up to (doing computing for just an hour a week means things are easily forgotten); it helps them to feel invested, empowered, and prepared (they have already solved the problem, now all they need to do is code it — this creates a massive boost in the engagement of girls in my lessons, but that's another story). Now imagine a project without this design stage: pupils will often get confused, or change their ideas and goals from lesson to lesson. This can mean that you as a teacher have less of a grasp of the direction their learning is taking, and you will have more fires to fight as they struggle to implement new ideas that have taken them off-piste.

An AFL opportunity

This is where the fantastic and unique assessment-for-learning opportunity comes in, which can have an immediate impact on learning. Consider the quiz algorithm in **Figure 2**, created by one of my Year 5 students (aged nine). She has designed her title screen, written her quiz questions, and highlighted the answers in orange. I then asked the class to highlight in green the aspects of their algorithms that they knew how to implement as code, and in pink, the areas they did not. As you can see, she went one step further and clarified her self-assessment with phrases such as, "I know how to make my sprite talk", "I don't know how to tell them they are right or wrong", and "I don't know how to get on to the new question." Now I have a whole class set of design algorithms, each indicating what they know and what they need to know.

My next move will be focusing on adapting my teaching to specific groupings of students. Some groups I can teach, some groups I can write instructions for, and some groups I can get my digital leaders to support (these are pupils whose role it is to further the development of computing in school). Together with a saved computer project, or a screenshot of their code, these algorithms provide an essential insight into the progression of skills and the learning journey that a child is on, together with a record of intervention from the teacher (if you are marking/commenting on their sheets) that has enabled them to implement their algorithm successfully. Pupils can then return to their algorithms when running the code, altering their designs in the evaluation and debugging stage, demonstrating further skills that can be assessed easily.

Next time you start a computing project, why not try getting the paper and pencil out before the laptops? Get the talk going, the ideas flowing, and the creative planning happening, and take time over this stage (I sometimes use two lessons to complete it thoroughly). Then ask your pupils to show you the bits they can or can't already code. You'll be surprised at the difference this will make to both their engagement and your assessment, not to mention their independence! (HW)

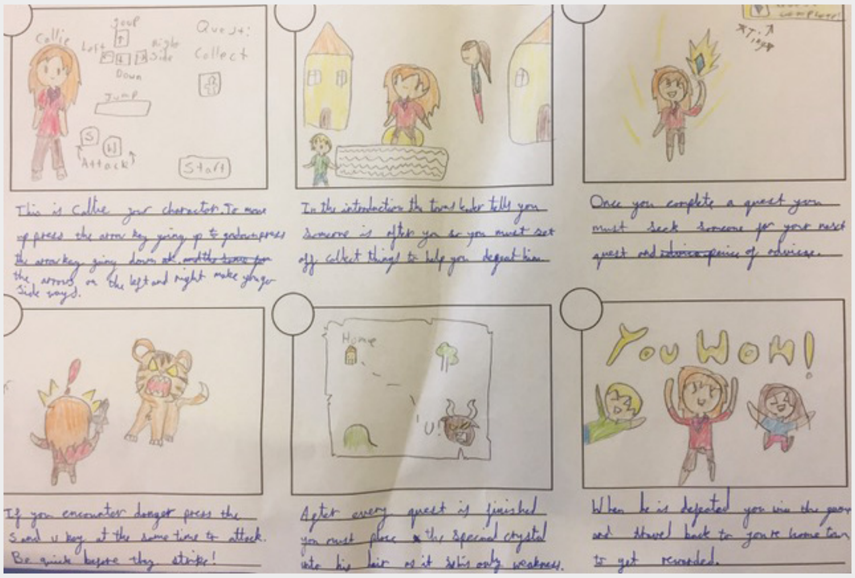

Collie
Left
Jump
Right Side
Down
Quest: Collect
Attack
Start
This is Collie your character. To move up press the arrow key going up to go down press the arrow key going down ok ~~and the arrows for~~ the arrows on the left and right make you go side ways.
In the introduction the town leader tells you someone is after you so you must set off collect things to help you defeat him
Once you complete a quest you must seek someone for your next quest and ~~advice~~ pieces of advice.
Home
You Won!
If you encounter danger press the S and W key at the same time to attack. Be quick before they strike!
After every quest is finished you must place the special crystal into his lair as it is his only weakness.
When he is defeated you win the game and travel back to you're home town to get rewarded.

Figure 1

> DESIGN ALGORITHMS HELP US FEEL INVESTED, EMPOWERED, AND PREPARED

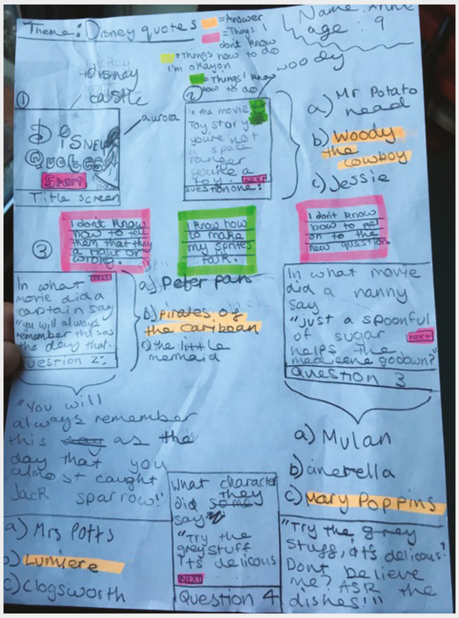

Theme: Disney quotes
= Answer
= Things I don't know
= Things I'm okay on
= Things I know how to do
Name: Anne
age: 9
Disney castle
aurora
Disney Quotes
Start
Title screen
In the movie Toy Story you're not a space ranger you're a toy.
next
Question one:
a) Mr Potato head
b) Woody the cowboy
c) Jessie
I don't know how to tell them that they are right or wrong.
I know how to make my sprites talk.
I don't know how to get on to the new question.
In what movie did a captain say "you will always remember this as the day that..."
Question 2:
a) Peter pan
b) Pirates of the Caribbean
c) the little mermaid
In what movie did a nanny say "just a spoonful of sugar helps the medicene go down?"
next
Question 3
"You will always remember this ~~day~~ as the day that you almost caught Jack sparrow!"
a) Mulan
b) cinderella
c) Mary Poppins
What character did they say
"Try the grey stuff it's delicous
NEXT
Question 4
a) Mrs Potts
b) Lumiere
c) Clogsworth
"Try the grey stuff, it's delicous! Dont believe me? Ask the dishes!"

Figure 2

WHITEBOARDS FOR ASSESSMENT

The CAS Assessment Working Group discuss the simple power of whiteboards for formative assessment

WRITTEN BY Tig Williams

TIPS FOR BUYING WHITEBOARDS

- ✓ Buy A3 whiteboards (or ledger size in North America). You may be able to manage with smaller boards for answering questions, but there are other uses for them in the computing curriculum (such as entity diagrams when teaching databases, or flowcharts in programming) and these need a lot of space!
- ✓ Even though they cost more, buy PVC-core boards. The MDF-core boards are a lot cheaper, but it is only a matter of time before the corners start to peel apart. Once this happens, little fingers usually finish the process quickly!

Mini whiteboards can be one of the most powerful tools in the computing classroom. They are not a solution for every issue, but they can be a surprisingly powerful way to help identify misconceptions quickly and increase participation from less confident students.

Gauging understanding

My favourite assessment-for-learning method using whiteboards is to pose a question as part of a lesson's mini plenary, to check learners' understanding of one of the lesson objectives. Give every student enough time to write an answer. This will obviously be longer for an open question, but it is important that every student writes their own answer. I then like to get all pupils to hold their whiteboards up at the same time. This method has a few advantages for formative assessment:

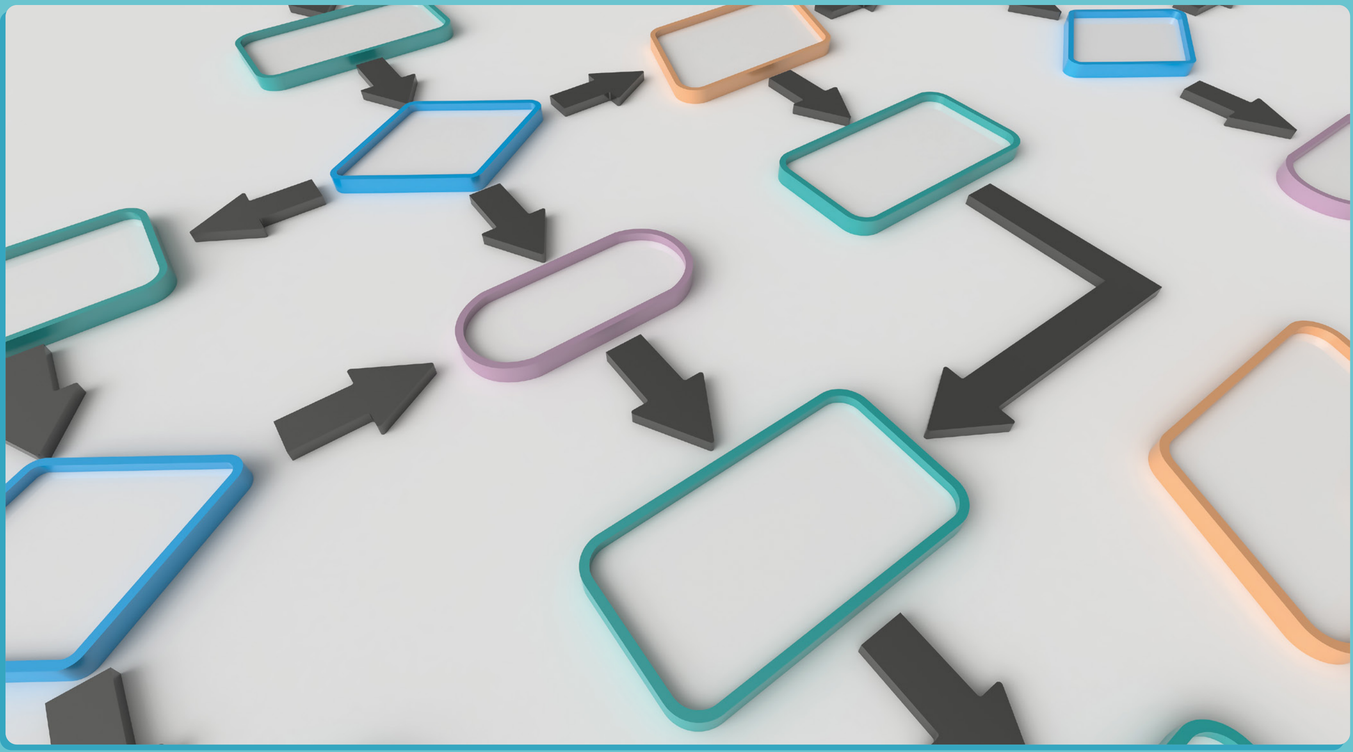
- If two students next to each other have the same answer, I can then make a judgement call about whether one has copied the other. In that case, the next question to one of the students would be, "Why do you think that?" This allows me to figure out whether they both understand, or if one is trying to avoid thinking!
- I can glance at the answers around the room and get a quick feel for the entire class's level of understanding. I can then make an informed decision to either reteach an earlier concept if understanding was not complete, or move on with the confidence that the class understands the basics.
- I can immediately identify students who have common misconceptions, and can then use questioning with the whole class to correct those misconceptions without singling out any student. It also means I can pick up misconceptions before I have finished teaching the topic. This allows me to stop the class before they have time to embed, which means we can spend time on additional content later in a student's learning journey, rather than on correcting those misconceptions.

A supportive tool

Whiteboards are also good for students with weak literacy skills or low confidence. These students are often reticent about putting anything on paper, where a mistake seems permanent; having the ability to immediately erase errors frees many students from this inhibition. Another benefit is that the whiteboard gives fidgeting students something to doodle on while they listen.

If we are careful to use questions that every student can answer, such as those testing understanding of a lesson's learning objectives, we can increase the engagement of the whole class. With the whiteboard approach, no student can sit back and allow others to field questions! We can also use the answers around the room to foster conversation about the topic, which increases understanding and interest for the whole class.

Though they may seem like a simple tool, whiteboards have a lot of potential for gauging misconceptions, bolstering students' confidence, and adding a little bit of fun to your classes! (HW)



EXPLORING ALGORITHMS

Josh Crossman discusses the importance of algorithms and exposing learners to their different representations

What comes to mind when you think of an algorithm? Many people may think of a list of text or a flowchart demonstrating steps that take place one after the other. These are both perfectly acceptable ways to lay out an algorithm, but it is important to expose young learners to a breadth of examples of algorithm representations. For example, the images in this article demonstrate a number of representations of different algorithms. In **Figure 1**, the algorithm comprises some symbols that show the sequential steps a learner needs to implement, while **Figure 2** shows the steps of an algorithm laid out as a flowchart that shows the use of selection. **Figure 3**, meanwhile, demonstrates an explanation of a program next to a pictorial representation, with additional information showing how the final program might look. All the examples look different in some way. Despite their differences, all the images demonstrate a representation of an algorithm. This article will outline what algorithms are and why they are important for young learners developing programs, before considering some ways to determine which representation would work best for a particular outcome.

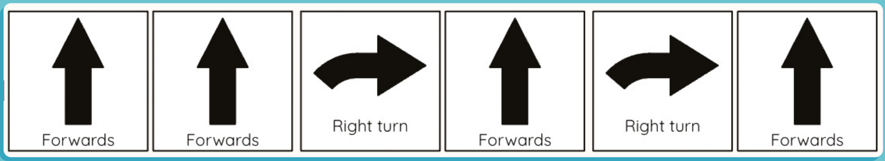


Figure 1 An algorithm represented by simple symbols

What are algorithms?

One thing to be mindful of when thinking about algorithms is the difference between an algorithm and algorithmic thinking. In many primary schools, teachers use unplugged activities to demonstrate the need for order and precision in algorithms, such as writing a recipe for making jam sandwiches, or creating instructions for drawing crazy characters (**helloworld.cc/crazycharacters**). These types of activity can help learners begin to recognise important aspects of algorithms. For example, they demonstrate the need for unambiguous steps and precise language — imagine putting the jar of jam on the bread! They can introduce sequencing to learners, as

Activities such as writing the instructions for making a jam sandwich help learners to recognise important aspects of algorithms

they begin to recognise the need for order, for example, taking the lid off the butter before using the knife. Or they can introduce decomposition by breaking down large tasks, such as focusing on just the head first when drawing crazy characters.

These activities definitely have merit to help learners think algorithmically; however, it is essential that the learning from activities such as those above is followed up with a code-based task to distinguish between algorithms and instructions.

So what exactly is an algorithm? And how is it different from instructions?

An algorithm can be described as a precise set of ordered steps to solve a problem or achieve a specific goal. For a computer to complete a task, the steps must be implemented in code. All algorithms will share some similarities, such as:

- The instructions are in a specific order
- Where relevant, the instructions have additional details, such as how many steps to move, to make them more precise and to prevent people from misinterpreting them
- The instructions can be implemented as code and followed by a computer

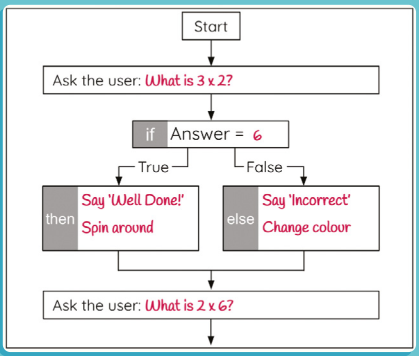


Figure 2 An algorithm represented as a flowchart, demonstrating selection

The importance of algorithms

We know how important programs are in the modern world. They are used in almost every aspect of our daily lives, from smartphones to cars, and from healthcare to finance. For programmers to create the programs at the core of these devices and industries, they first have to understand how they want a program to work. Experts might be able to do this in their heads (experience and practice give them lots of examples to refer to), but for younger learners, thinking both about how you want the program to work **and** how you want it to do it, is too much.

To reduce the cognitive load you are placing on students, you can abstract programs so that they can focus on just one thing at a time. Abstraction is a skill that expert programmers routinely use, adjusting their focus while developing a programmed solution. It includes being able to move between the specific goals of a task, the design for a solution, and the building and coding of a program, to how that program behaves when it is run. These different perspectives can be modelled through the levels of abstraction (LOA) model (**helloworld.cc/LOA**):

- **Task:** the task level outlines the problem, or describes what a project should do when it is run.
- **Design:** the design level is made up of a number of different parts. These include the algorithm and other project components, such as artwork and sounds. This part of the project will be more detailed than the task, but will not refer to the code of the project.
- **Code/build:** the code level (or build level, for physical computing projects) represents a program that implements the design level, including building a physical computing project if necessary.
- **Run the project:** this level focuses on how the program works when it is run.

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.

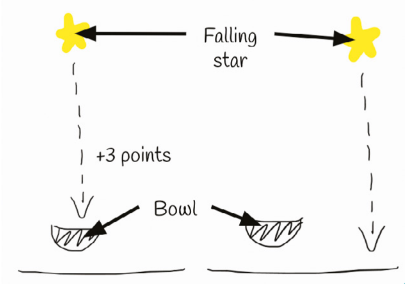


Figure 3 An algorithm comprising text and a pictorial representation

This approach doesn't represent a linear path. As learners work on a program, they will repeatedly switch between levels as they implement code, debug, and return to their designs sporadically. Typically, learners spend most of their time at the code/build level. However, it is important to ensure learners are given the time they need at the design level too. Creating an algorithm that outlines what they want their program to do, and sketching out how they want it to look, will enable them to create programs that are achievable, and ensure that the cognitive load is kept at an optimal level. You can find more information on the LOA approach in *The Big Book of Computing Content* (**helloworld.cc/bigbook2**).

> ALGORITHMS ARE USED IN MOST ASPECTS OF OUR DAILY LIVES, FROM SMARTPHONES TO CARS AND FROM HEALTHCARE TO FINANCE

Choosing an appropriate algorithm representation

As algorithms can be created in many different formats, it can be difficult to decide on the best way to represent one for a program. It is important to recognise that there is no right way, but there are some considerations to bear in mind that might help you decide which representation would suit a particular expected outcome:

- **The experience of the learners:** if learners are at the beginning of their programming journey, you might consider ensuring that they experience simpler algorithms, such as symbols (e.g. **Figure 1**) or linear steps. This can begin to expose learners to algorithmic thinking and provide them with an understanding of how algorithms can be used to structure code.
- **Program genre:** some algorithmic representations are better suited to different genres. If a learner is creating a game that requires a lot of repetition, for example, this will need a representation that can show the repetition clearly. This could be shown by introducing indentation into a set of linear steps, to demonstrate the need to repeat a section.
- **Program complexity:** similarly to program genre, the complexity of the program the learner is trying to create needs to be considered. For example, selection within an algorithm can be shown more easily through a flowchart representation (e.g. **Figure 2**) than by a list of symbols.

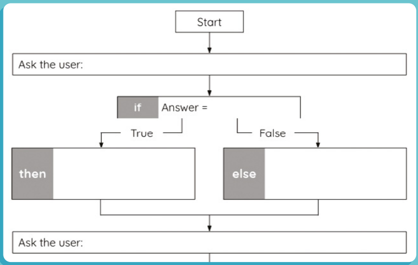


Figure 4 A part-completed algorithm

Another important thing to think about is when to mandate how an algorithm is represented, and when to let the learners decide for themselves. Some of this will depend on the learners' experience, and the complexity of their programs. For example, if all learners are working on a floor robot program, mandating the use of a symbol algorithm makes sense, whereas a learner with more experience who needs an algorithm for a sprite in a larger program might be given the option to choose an algorithm representation that suits the style of their desired program. Equally, you might give learners examples or part-completed algorithms (e.g. **Figure 4**). This models to learners what an appropriate algorithm might look like, and enables them to focus on a specific section of an otherwise larger algorithm. After all, you don't know what you don't know!

Think about your own school and the computing curriculum you use. How are algorithms introduced to learners, and are the learners given opportunities to see a range of algorithm representations? Are they given a fair amount of time to develop their program design before jumping into the code? Is the importance of algorithms and the way they support coding discussed with the learners? If you want more ideas and examples of algorithms, you can download the Raspberry Pi Foundation's Computing Curriculum at **helloworld.cc/tcc**. It's free and introduces learners to a progression of algorithm representation for novice learners. (HW)

JOSH CROSSMAN

Josh is a programme coordinator at the Raspberry Pi Foundation, working across programmes such as The Computing Curriculum and Hello World.

Expanding children's use of technology can encourage them to explore a wider range of skills

HOW ARE YOU PREPARING YOUNG CHILDREN FOR A DIGITAL WORLD?

Sway Grantham explores the importance of technology in the lives of our youngest learners

Technology use with our youngest learners is a hotly debated topic. From governments to parents, from learning outcomes to screen-time rules, everyone has an opinion on the right approach. However, in 2023 in the UK, Ofcom reported that 87 percent of three- to four-year-olds and 93 percent of five- to seven-year-olds went online at home (helloworld.cc/ofcom2023). Children are clearly exposed to technology use as a part of their world at home, and school should be no different.

As educators, we have a responsibility to prepare our learners for life in a digital world. We want them to understand its role, be aware of its risks, and have the wide range of experiences that would not be available without it. Beyond this, we need to consider the other 13 percent of three- to four-year-olds and the other 7 percent of five- to seven-year-olds, who have not experienced technology at home. School should be the great equaliser, and it is our responsibility to ensure our youngest learners have access to the digital skills they need.

Exploring the wider world

A huge part of early years or kindergarten education is about learners sharing their world with each other and discovering that everyone has different experiences and does things in their own way. Technology is a huge part of this. Allowing learners to share their personal experiences of using technology both accepts the permanent role it has in our lives and introduces learners to the broader role of technology in helping us to learn, talk to people, have fun watching videos, and do a job.

Many learners will spend time primarily using technology to do just one of these things. Expanding their use of technology can encourage them to explore a wider range of skills and show them how to use technology in different ways. These explorations can appear within their environment, such as within the role play area of a classroom, where learners can use a range of toys to demonstrate how they have seen technology being used in their lives. This non-technical use of toys may seem counter-intuitive to reducing the digital divide, but if you don't know what technology can do, how will you ever go about learning to use it? There is also a range of digital role play apps (such as the Toca Boca apps, **tocaboca.com**), which can allow learners to recreate their experiences in real-world situations, such as the hospital, a hair salon, or an office. These are great opportunities to extend role play areas beyond the resources you already have.

Another aspect of a child's world that technology can facilitate is their understanding of the world beyond their home town. Technology allows learners to explore the wider world and follow their interests in ways that are largely impossible without technology. Take virtual reality experiences such as the ExpeditionsPro app (**expeditionspro.com**), where learners can explore Antarctica or even the bottom of the ocean, or augmented reality experiences such as Octagon Studio's 4D+ cards (**helloworld.cc/octagon**), which allow you to see sea creatures and safari animals popping out of your screen. You could do a joint project with a class of children in another country, in which learners blog or share 'emails' with each other. Each of these opportunities gives children a richer understanding of the world while also using technology in meaningful ways.

Technology as a tool

Beyond helping children to better understand our world, technology offers opportunities to be expressive and imaginative. For example, how about using an app such as Draw & Tell (**helloworld.cc/draw&tell**) alongside your art activities? This drawing app offers learners multiple ways to draw pictures, and then allows them to record themselves explaining to you what they are drawing. Or what about using filters on photographs to make artistic versions of themselves, their pets, or their favourite toys? Technology should be one of a range of tools learners have access to for creative play and expression, particularly as they offer opportunities that would be impossible in more analogue ways.

“ CHILDREN CAN USE TECHNOLOGY TO BETTER UNDERSTAND OUR WORLD, FOR CREATIVE PLAY, AND FOR EXPRESSION

Using technology is also invaluable for learners who struggle with their communication and language skills. When you find speaking challenging, it can often be intimidating to talk to others who speak more confidently. But speaking to a tablet? A tablet only speaks as well as you do! Using options to record sounds and listen back to them is an excellent way for young children to learn how clear their speech is and to practise speech exercises. ChatterPix Kids (**helloworld.cc/chatterpix**) is a great app for this. You take a photo of something like your favourite soft toy and then record yourself talking. When you play it back, it looks as if the toy is saying your words. This is an incredibly engaging way for learners to practise communicating.

No matter how we feel about technology's role in young people's lives, it is a part of their world. We need to ensure that we are giving learners opportunities to develop digital skills, recognise the role of technology, and understand how it can be used for social good. It is not just about preparing them for subsequent computing education (although this is definitely a bonus!) or about online safety (although this is vital — see my articles in issues 19 and 20 of Hello World for more about this). It's about their right to be active citizens in the digital world. So I ask again, how are you preparing young children for a digital world? (HW)

SWAY GRANTHAM
Sway is a senior learning manager at the Raspberry Pi Foundation, where she leads the educator development team developing resources and training for computing teachers around the world (**@SwayGrantham**).

LEADING PRIMARY COMPUTING

Neil Rickus discusses how guidance from Computing at School is supporting primary computing leads and explores some reflections from the community

As a subject, computing is vital for giving all children the knowledge and skills they need to succeed in our digital world. It is an essential subject within the school curriculum through its contribution to children's personal and intellectual development, their understanding of the world, and their future education and employability. Primary subject leaders for computing are vital catalysts for change in schools across the globe. It is under their leadership that every young person can begin to thrive within our digital society.

Leading computing can be daunting, especially for teachers without the subject knowledge. However, with support and encouragement from other subject leaders, teachers can feel empowered, equipped, and excited about the opportunity they have to shape the experiences of the young people in their care. This is exactly the aim of the *Computing at School (CAS) Leading primary computing: reflections from the subject leader community toolkit* (**helloworld.cc/primarytoolkit**) resource, which outlines case studies and recommendations from active practitioners across the computing community. Although the toolkit was developed in the UK, it can be used by schools globally, and most of the case studies and guidance are applicable to curriculum content around the world.

CASE STUDY EXAMPLE: FAMILY AND COMMUNITY ENGAGEMENT

Will Franklin: computing coordinator, Westdene Primary School, Brighton, UK

"As part of engaging families with the computing curriculum at Westdene, we run a number of different events, including:

- Online safety workshops for parents and carers, focusing on the technologies children typically use and how they can use them safely
- Coding competitions during Hour of Code (**hourofcode.com**), in which parents and carers are invited to take part and compete against each other
- Support sessions on how to use our online learning platform, including how to view work, access online resources, and submit completed assignments

"To engage with our local community, we collaborate with local schools to run Kidsmeet sessions, at which digital leaders from each school come together to take part in joint computing activities. This also provides an opportunity for the children to share their expertise with one another and examine how computing can be developed within the school. Finally, we also regularly reach out to local industry, which has involved partnering with MakerClub to run a school-based version of their popular after-school technology club."

Toolkit development and aims

The toolkit was developed in conjunction with primary teachers and computing education experts to highlight good practice within schools when leading the subject. Throughout its development, CAS recognised that computing in primary schools is often led by non-specialists, so they took care to ensure that the guidance outlined is accessible and appropriate for teachers with limited computing experience and/or formal qualifications. For example, they sought case studies from those with significant experience in supporting non-specialist teachers in schools, with explanations of technical vocabulary and concepts where appropriate.

The guidance aims to enable primary computing leads to develop their pedagogical and subject knowledge by engaging with appropriate learning opportunities. It spans the following areas:

- Computing subject leadership
- Curriculum
- Assessment
- Resourcing
- Enrichment
- Family and community engagement
- Diversity and inclusion
- Online safety

Subject leads can use the toolkit to help examine strategies for evidencing pupils' attainment; to increase their awareness of technology used by both pupils and parents; and to give them demonstrable expertise in the subject. The toolkit should also help subject leads to support other staff in the school in delivering excellent computing lessons and implementing appropriate cross-curricular links, while also enabling children to use technology safely and responsibly. Equipping subject leads and other teachers with these tools can then enable schools to develop their policies and procedures, such as those for online safety and safeguarding, alongside accessible curriculum content. This then ultimately supports pupils in making excellent progress within engaging computing lessons.

CASE STUDY EXAMPLE: LEADING AND DEVELOPING STAFF

Stella McCarthy: head of school and digital strategy lead, Dame Tipping Church of England Primary School, UK

"The key to leading the introduction of new technology in any school is ensuring teachers receive regular, high-quality CPD. Staff CPD needs to relate to both the tools available and to the associated pedagogy. In particular, through encouraging teachers to think about how they can use the tools and software available to them, they can enhance learning and teaching, plus see the benefits in the classroom.

"In addition to this, as computing coordinator, it is important you don't purchase or introduce anything new without being willing to learn how to use it effectively; then, once you have mastered it, be able to train all staff — preferably in small steps, with hands-on opportunities. Once appropriate hardware and software are in place, you can embed the use of technology across the whole curriculum. Effective CPD can therefore lead to intuitive use of technology for computing and across the whole curriculum."

SUBJECT LEADERS CAN HELP TEACHERS TO FEEL EMPOWERED, EQUIPPED, AND EXCITED TO LEAD COMPUTING

Using the toolkit

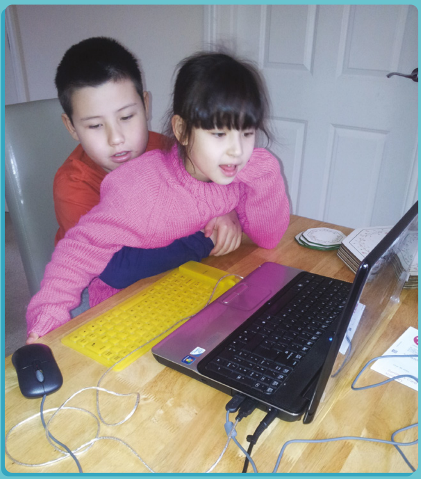
Depending on your school's current computing provision and the experience of the subject lead, you can use the guidance and case studies in a number of different ways. For example, new subject leads, or schools wanting to review their whole-school provision, might begin with using the complementary NCCE Computing Quality Framework (**computingqualityframework.org**) to review the school's existing computing provision. This will then enable them to identify focus areas linked to their curriculum and the school's priorities. The subject lead/school can then refer to the recommendations and case studies related to those focus areas and put development plans in place for their school as required. Finally, they can examine and record progress against each target they set for themselves in their development plan, such as every half term, with further focus areas looked into as appropriate. You can find a couple of examples of case studies in the boxouts, to get a flavour of the kind of support the toolkit offers.

Have you used the CAS toolkit? Did you find any of the case studies useful? I'd love to hear from you — get in touch via **@computingchamps** on Twitter. (HW)

NEIL RICKUS
Neil is a senior lecturer in computing education at the University of Hertfordshire, UK. He also works at UCL and Brunel, and is a primary education consultant for the BCS and the founder of Computing Champions. He is a CAS community leader, a CEOP Ambassador, an FHEA, and a Raspberry Pi, Google, and Microsoft Certified Educator.

CODING: A STUDENT'S PERSPECTIVE

Student **Carmella Abela** shares how she got started with coding and the elements of programming that particularly captivated her as a child


The first time my brother showed me the game he made!

When I was seven, my older brother showed me this incredible game he had made in Scratch, in which you drove a car around a race track. That's where my interest in coding started. Looking back, it wasn't really the game itself that made coding seem so amazing; it was the fact that someone just a few years older than me had made it. I think that's the key element that makes a kid pursue programming: the fact that they themselves have the potential to create something unique.

Despite my interest and amazement, I was still reluctant to get started. My general idea of coding had always been some stereotypical nerd typing away lines and lines of code. This sounded incredibly boring, not to mention hard. However, once I got started, I was pleasantly surprised to discover that coding can be an enjoyable activity — especially when using visual programming languages, such as Scratch or Blockly on Code.org.

Feeding your imagination

I've always preferred Scratch, as it allows more freedom and choice to make anything you want. The first game I ever made, aged eight, was called Car Jump, in which a car would jump to dodge obstacles coming its way. Slowly, I learnt to implement variables such as lives and points, and eventually, continuously moving lines to make the car seem as if it was travelling. As my coding skills developed, I made more complex platform games such as versions of Mario, Geometry Dash, and Pac-Man. Obviously they weren't as impressive as the originals, but the pride I felt when I saw the final products made my day. Scratch's visual nature means you can see the results of your code right away, which was very satisfying and rewarding for a young coder like me!

The most fun aspect of using Scratch was being able to design and develop characters that could move and talk. This was especially memorable as it allowed me to personalise each project I made. I spent hours drawing characters for each game, and each time it was very enjoyable. The characters represented me, my friends, or anyone that would fit into a specific setting. By bringing the sprites to life, I could really use my imagination because they could actually move and talk! In Scratch, I was able to create animations, games, and interactive stories using these characters. And I could customise my projects with graphics, sound, and other media, which made the process more engaging and entertaining.

I also enjoyed using platforms such as Code.org, which encourage experimentation and offer immediate feedback, turning coding into more of a game than an assignment. It teaches young children to code using well-known games such as Minecraft, motivating us to continue playing the game without even realising we're learning. Furthermore, after finishing courses, we received badges, certificates, and other rewards, which gave me a sense of accomplishment and an incentive to do more.

I thought that, looking back, all the games I made years ago would be terrible; but while there were some odd ones, I also discovered a few that were quite impressive. My favourite game is a six-level interactive story game that gets progressively harder as two children travel through space, trying to rescue their parents. Not bad for a nine-year-old! My advice to teachers would be to give students the freedom to explore when they are learning to program — you never know, maybe they'll end up creating the next big game! (HW)

CARMELLA ABELA
Carmella is 16 years old and is currently doing her IGCSEs. She has been entering coding competitions since the age of nine.

A FOUNDATION FOR FUTURE SUCCESS

Sethi De Clercq explores the role of computing education in lower-primary schools and looks at why we need to start learners young

Starting early with a computing education is crucial for building a solid foundation of computational thinking skills and fosters long-term benefits such as children's career-readiness and making the STEM workforce a more diverse and inclusive place to be.

Vital computational thinking skills

Research shows that early exposure to computational thinking can help children develop problem-solving, critical thinking, and logic skills that are useful in all areas of life, not just in computer science. For example, a study by the University of Canterbury found that children who learnt to code early showed improved creativity and problem-solving skills, plus increased confidence and persistence when facing complex tasks (**helloworld.cc/bell2009**). I have experienced this in my classroom, where unplugged debugging activities contributed significantly to the children's can-do attitude, even if this meant making mistakes. Another study found that teaching computational thinking to primary-school children can enhance their ability to reason, analyse, and improve their mathematical skills (**helloworld.cc/myers2014**).

Even if you don't have huge amounts of space in your timetable for computing, you can still incorporate computational thinking and digital literacy activities into your classroom's continuous provision set-up, which children can access independently throughout the day. In my classroom, we have a role play station with laminated keyboards to familiarise our youngest with everyday tech. The students colour in the keys with whiteboard markers to spell out our weekly common exception words taught during phonics lessons. The construction corner also has algorithm cards that have various building instructions with increasing degrees of complexity, which children can try out with the available Lego, Duplo, and wooden blocks. Finally, the ScratchJr tiles have also found their way into our small-world area (settings made to invoke talk and play) as laminated printouts for students to use as prompts for sequencing their role play.

SETHI DE CLERCQ

Sethi is the teach-lead for computing in Pre-Prep and head of Key Stage 1 at Rugby School Thailand. He leads professional development on educational technology, and shares his experiences on YouTube (Flipped Classroom Tutorials) and Twitter (**@SethiDeClercq**).

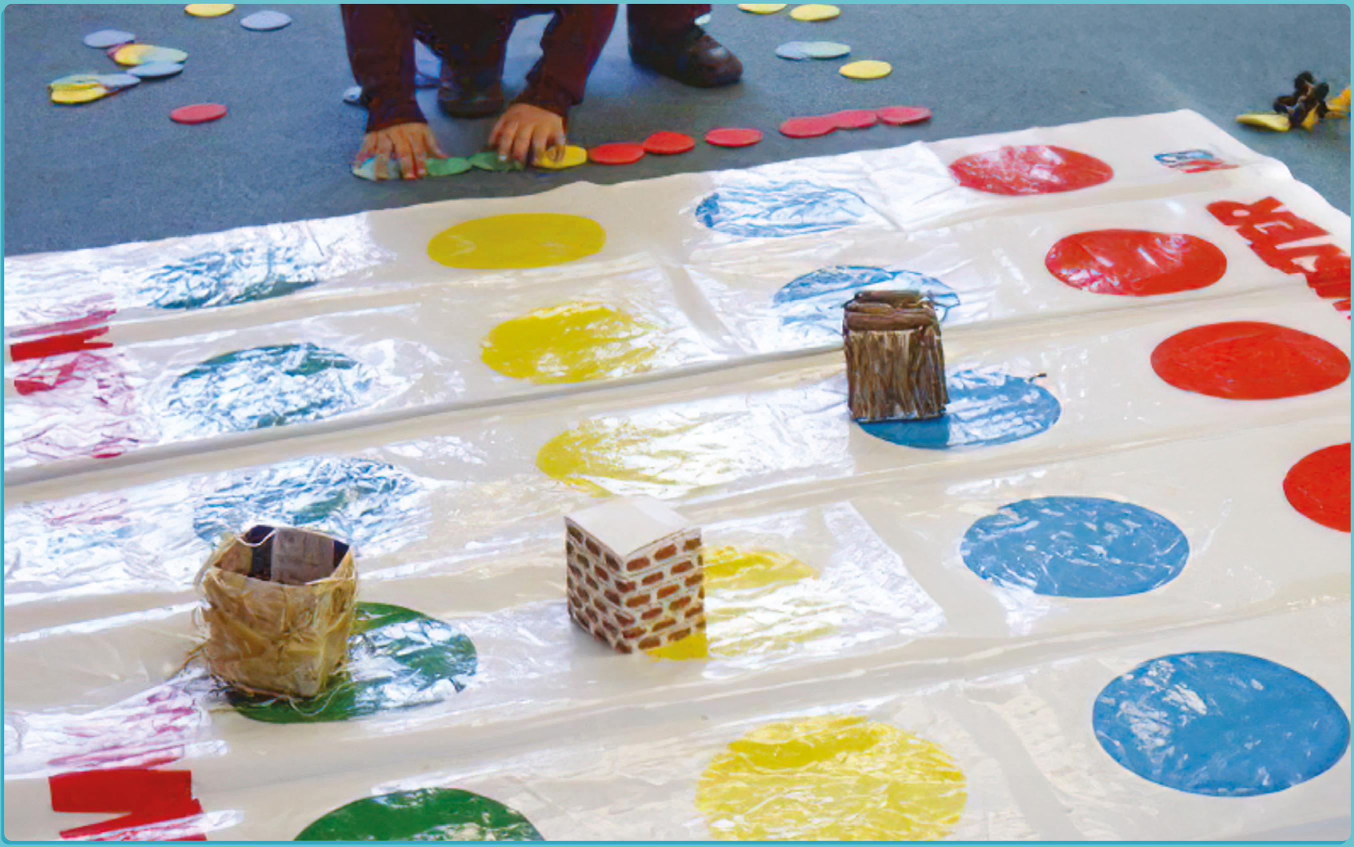
Working life

Starting early with computing education also has longer-term benefits. In today's digital age, many jobs require at least a basic understanding of technology, and this trend will only continue. By giving children a robust set of computational thinking and digital literacy skills, we can help them prepare for the workforce of tomorrow, which will include many jobs not yet in existence.

Moreover, computing education can help bridge the gender gap in technology fields. Studies repeatedly show that girls are less likely to study computer science in higher education and pursue careers in technology (take a look at the articles on pages 10 and 44). However, introducing computing education to our youngest learners can help break down stereotypes and inspire girls to pursue computer science. Having positive role models from all walks of life within these fields also contributes to making technology a viable study choice for all.

For example, we recently looked at significant people throughout history. We highlighted both Hedy Lamarr, who developed technology that formed the basis for today's Wi-Fi, and Katherine Johnson, a mathematician at NASA. Lamarr led us on a journey around the school looking for any wireless technology that may be used, and an old-school AM/FM radio found its way into the role play area. With Johnson, the children were inspired as they used a map to plan a simple route from one place to another. They talked about how maths is used to measure distances and plan trajectories, and we discussed how Johnson's calculations helped get people to space and back safely.

Computing is an essential part of the curriculum and will only become more important. By introducing computing and its key skills to our youngest learners, we can help them develop problem-solving, critical thinking, and logic skills that will serve them and the future STEM workforce well. (HW)



Children gave instructions for the wolf to get from the house of straw to the house of bricks

PICTURE AN ENGAGING COMPUTING LESSON

Rachael Coultart shares how she has used picture books as starting points to deliver engaging computing lessons and embed programming concepts in her classroom

My passion for reading stories to children has been a cornerstone of my teaching career. Captivating pupils' imaginations and exposing them to rich vocabulary, I've found that picture books always provide exciting starting points for a wealth of classroom experiences. So when I became computing subject lead (despite my technophobic tendencies) and had to rapidly upskill myself, it was reassuring to discover that stories could help me embed some early programming concepts.

The more I learnt about programming, the more I began to see overlaps with other subjects, such as geography (sequencing a journey) and English (writing instructions). It was during a discussion on Twitter that **@BradleyDardis** shared an idea he'd used in the classroom based on Julia Donaldson and Lydia Monks' *What the Ladybird Heard* (**helloworld.cc/ladybird**). He had children draw a large-scale map of the farm in the story and control a Sphero (dressed as a ladybird) to retell the story by driving it from one place on the farm to another.

I tried this activity with the five-year-olds in my class, and the results were amazing! There was the inevitable excitement over just making it move for the first time and seeing how fast it could go. However, the overall levels of engagement from both boys and girls in drawing the map, retelling the story, and controlling the Sphero were far greater than I had seen in similar activities. As the children worked together in groups, they were keen to remind one another about where the ladybird had to go next and repeated phrases from the story, such as "pass the horse and then turn right" and "through the door of the fine prize cow's shed".

Stories provide different contexts

So began my exploration of stories for teaching programming concepts to young children. I started by developing the idea of an algorithm being a precise sequence of instructions through fairy tales such as 'The Three Little Pigs' and having children give the wolf instructions to get from the house of straw to the house of bricks. They did this physically at first, directing one another around a Twister mat. Then we progressed to giving a programmable robot (dressed up as a wolf) the instructions to get from A to B.

The more stories we read, the more contexts we found for practising giving and receiving instructions, and creating algorithms and turning them into code to retell a story or adapt it to make it our own. Whether it was being the mouse and meeting various creatures in Julia Donaldson's *The Gruffalo* (**helloworld.cc/gruffalo**) or eating lots of food in Eric Carle's *The Very Hungry Caterpillar* (**helloworld.cc/caterpillar**), children persevered with getting their algorithm right, collaborated to debug their code, and demonstrated an increasing level of skill in their programming abilities as the year progressed.

Young children have an enormous propensity for imaginative storytelling, and I've found that stories help them to grasp new ideas. Stories help us make sense of the world. They connect pieces of information together in meaningful and memorable ways. Through adopting this approach in computing lessons, I found learners were more readily able to take learning from one lesson and context to another. One of the outcomes I found most exciting was that when I introduced a new programmable robot, children instantly looked to see how to make it move forward and turn. When I left it out for them to explore in their own time, they started making up their own stories for it and making outfits to stick on it!

Exploring the research

I've recently had the opportunity to be involved in some classroom-based research looking at gender balance in computing, using a storytelling approach to engage girls (**helloworld.cc/GBICstory**). This was conducted with my class of seven-year-olds using ScratchJr to introduce programming principles, such as sequencing and repetition, through digital story-writing. Using this approach was the first time I had the whole class achieve success in programming, irrespective of gender or ability level — affirmation that this storytelling approach was working for me and my learners!

I've also enjoyed delving into Bobby Whyte's research about using a storytelling approach (see pages 16-17). He concludes that we have to carefully consider the implications of delivering cross-curricular literacy and computing lessons, and ensure that we don't overdilute the learning intentions of the other curriculum area. I've always been an advocate of cross-curricular computing, but research about culturally relevant pedagogy in computing has led me to consider this in a different light (**helloworld.cc/culturalrelevance**). It has led me to think that starting with what is important (culturally relevant) to our learners may be more important than linking computing to other subjects, or using other subjects to provide a context for computing. So whether it is *Paw Patrol* or Formula One that most excites learners, starting here brings instant engagement, and with instant engagement comes great learning.

If you would like to try this out in your setting, I would advise starting with a story that your children know and love. It's useful to look for a story in which the main character is trying to find something or go somewhere, as this lends itself to teaching programming concepts. If it involves a journey (like the fairy tale 'The Three Billy Goats Gruff'), you could link it to geography map work. If it involves looking for something (like Nick Sharratt's *Shark in the Park*), you could link it to positional language in maths. If you would like more starting points for using stories, you can access a resource I have shared at **helloworld.cc/storyideas**. How might you turn your favourite story into an engaging computing lesson? (HW)

RACHAEL COULTART

Rachael is a computing subject lead at St Nicholas Primary School in Stevenage, UK. She is an early years enthusiast with over 30 years of classroom experience teaching three- to eleven-year-olds (**@rcoultart**).

Stories help learners to grasp new ideas

GETTING STARTED WITH PHYSICAL COMPUTING

Graham Hastings offers a toolkit for getting started with physical computing in the primary classroom

When I got hold of my first micro:bits, I created a simple classroom demonstration that made it appear as if I could guess what image each of the children's micro:bits was displaying. They were amazed that I guessed correctly every time, and believed I could read the 'mind' of the computer. The class was immediately hooked and eager to learn more about this 'magical' device.

Physical computing like this involves designing and creating simple computer-based interactive systems that can sense and respond to the world around them. In practical terms, it is the connection of simple electronic components to a computer to input data and output some form of response. Examples seen in primary schools include an alarm that sounds when an object is moved, an automated lighthouse, a battery tester, a pelican crossing, a soil moisture sensor, a moving robotic arm, a computer-controlled buggy, and a radio-controlled light switch. In this article, I will take you through getting started with physical computing in the primary classroom, from why you should incorporate it into your computing lessons to recommendations for hardware and upskilling.

Why teach physical computing?

Yes, the topic is specified as an attainment target within England's national curriculum for computing at upper primary (and probably in your local setting, too). That said, there are better reasons for adding physical computing to your schemes of work. With the exponential growth of internet-connected devices that monitor and control our environment, it is important to engender in children an awareness and understanding of the interactive systems that sense, respond to, and control aspects of their world (the internet of things).

Physical computing is an excellent context to broaden and enliven children's computing experience. I think it is the perfect vehicle for introducing STEM projects with technology at their core. A well-designed project combines designing and making physical artefacts with aspects of maths, engineering, sensing, and control. I've found that children are frequently astonished by what they have been able to achieve, leading to feelings of immense satisfaction, pride in their work, and positive feelings towards computing.

Overcoming challenges

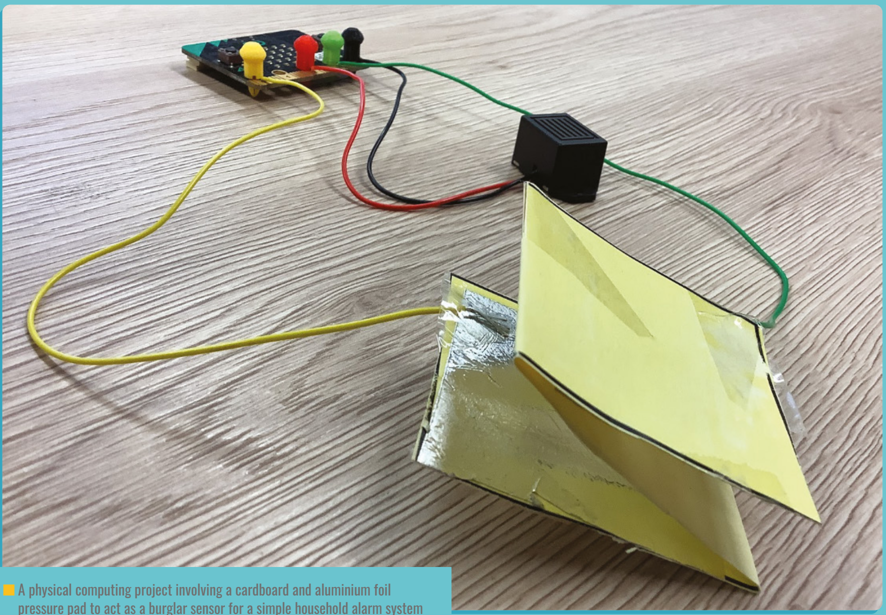
An initial barrier to introducing physical computing, particularly among teachers without a technical background, is a feeling that they lack the prerequisite knowledge and skills. The good news is that there are now many excellent and affordable resources designed for use with young children, and all the electronic components and connections for primary-school projects, such as push switches, LEDs, and buzzers, are fairly basic. The first project we do with our Year 4 children (aged eight to nine), for example, is to make a pressure pad from cardboard and aluminium foil to act as a burglar sensor for a simple household alarm system (pictured).

As a first step, I'd recommend thinking about the hardware that you want to use in your classroom. There are a number of devices suitable for physical computing, such as the Arduino, Crumble, micro:bit, Raspberry Pi, and Pico. The Arduino, Raspberry Pi, and Pico are better suited to secondary-school projects, so I'm just going to focus on the micro:bit and Crumble. Both devices have their own strengths and weaknesses (see the table opposite); the best choice depends on your preferences and the specific needs of your school. My school adopted the micro:bit over the Crumble because we like to make use of the radio feature to link micro:bits.

If you're nervous about getting started, there are several fantastic professional development options out there. My first

RECOMMENDATIONS

- Use this article's self-study links to learn more about physical computing in the primary curriculum and decide which device you would prefer to purchase for your school.
- Buy enough equipment for your class to work in pairs.
- Give your hardware and components to a small group of your more computer-savvy pupils. Point them to some online tutorials and facilitate them to learn, by experimentation, how to code the devices.
- Ask your pupils to teach you what they have learnt. When you introduce the equipment to your class, these pupils will become much-needed extra pairs of hands — your digital helpers!
- Start with a simple project with a context that your pupils can relate to and will understand, such as a domestic burglar alarm or a radio-controlled light switch.



A physical computing project involving a cardboard and aluminium foil pressure pad to act as a burglar sensor for a simple household alarm system

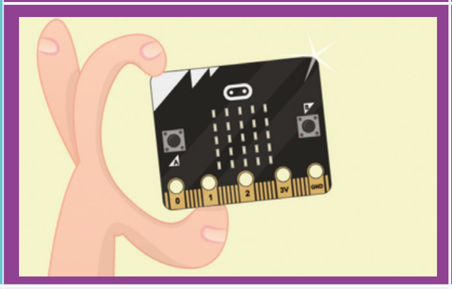
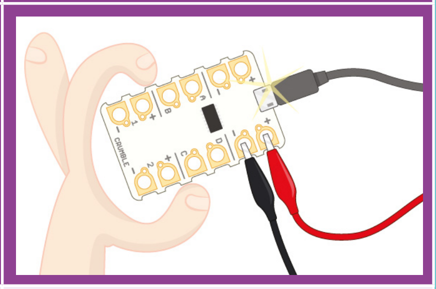
GRAHAM HASTINGS

Graham is head of computing at St John's College School, Cambridge, UK.

stop is always YouTube, where there is a vast collection of excellent videos explaining every aspect of physical computing using the micro:bit and the Crumble. Many of these have been created by teachers for teachers.

There are also plenty of self-paced courses out there that are hardware-specific. For example, you can find projects for getting started with the micro:bit at **helloworld.cc/mbitprojects1** and **helloworld.cc/mbitprojects2**, and the same for the Crumble at **helloworld.cc/crumbleprojects1** and **helloworld.cc/crumbleprojects2**. It's also worth checking out forums for educator discussions about trying these things out — for example, the Computing at School forum (**helloworld.cc/CAS**) or Redfern's Crumble-based discussions at **helloworld.cc/crumblechat**. If you can find a face-to-face course, even better — if you're a UK-based educator, there's STEM Learning's *Introduction to Micro:bit* course (**helloworld.cc/STEMmbit**) or Teach Computing's Crumble short course (**helloworld.cc/TCcrumble**).

Of course, the micro:bit is not magic, as my pupils first thought. It is just a very clever little piece of technology. The real magic is the visceral excitement in the classroom when children learn to program micro:bits and Crumbles to do amazing things as they measure and control the physical world. (HW)

	MICRO:BIT	CRUMBLE
		
Programming environment	Block-based code, web-based programming environment. Scratch has micro:bit blocks to incorporate micro:bits into Scratch programs.	Block-based code, downloadable programming environment.
Key features	A 5x5 LED display, wireless communication capabilities, sound, and built-in sensors enable an extensive range of potential projects.	A 4.5 volt battery pack gives it an edge for motor control as no additional control board is needed.
Teaching resources	Lesson plans, tutorials, and a range of project ideas with video tutorials are available on the MakeCode website (**helloworld.cc/mbitprojects1**).	Lesson plans and tutorials are available on the Redfern Electronics website (**helloworld.cc/crumbleprojects1**).
Peripheral devices	Cheap electronic components are available from various suppliers such as RS (UK-only: **helloworld.cc/RS**) and Farnell (**farnell.com**). Kitronik (**kitronik.co.uk**) stocks a number of components designed specifically to extend the range of the micro:bit.	Cheap electronic components are available from various suppliers such as RS (UK-only: **helloworld.cc/RS**) and Farnell (**farnell.com**). You can connect micro:bits and Crumbles with micro:pegs (Tinkercad file for 3D printing available at **helloworld.cc/peg**).

THE MAGIC OF TOUCH-TYPING

Nicki Cooper explains why keyboard skills should not be overlooked and why learning to type is a fun, engaging, and valuable skill for children

Over the last year and a half, I've had the pleasure of volunteering at my local primary school, with computing as my focus. Although it's not explicitly mentioned in England's national curriculum, I wanted to expose the children to keyboard skills and learning to type properly. I used to work as a computing secondary-school teacher, and was often frustrated when children arrived from primary school with very limited knowledge of how to use a keyboard. This was my chance to make a difference and help them explore touch-typing.

Developing typing skills is essential in today's world. Many careers will involve some element of typing on a computer, even if it's just to send emails. Having a good understanding of where the keys are, and being able to type quickly, will have obvious benefits. If children learn to touch-type from a young age, they won't have time to first build up the bad habits that will ultimately slow them down.

The magic of touch-typing

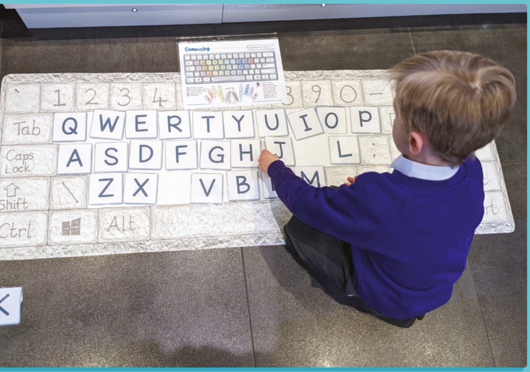
When I first introduced touch-typing to the children in both Year 1 (aged five to six) and EYFS (early years foundation stage, up to age five), I typed a sentence to display on the screen, purposely not looking at the keyboard while doing so. To my surprise, the children thought it was magic! One child was so impressed with this 'magic' that she relayed it to her grown-up when she bumped into me in a coffee shop a week later!

To get started, we worked on keyboard skills, by first discussing the different parts of the keyboard. With EYFS children, I displayed the keyboard on-screen, showing just the lower-case letters, and I added numbers and the Backspace and Space bar. With Year 1 children, we also talked about the purpose of the Shift keys, as they were more familiar with changing between upper-case and lower-case letters in their writing lessons. The first activity, for both year groups, was then colour-coding the keyboard. Their task was to colour each of their fingers in a different colour and then colour the corresponding keys that those fingers would touch in the same colour on their worksheets. This would help them to understand which fingers should be used on each key. Once they had coloured their worksheets, they had a go at finding the letters from their name and placing counters on the correct keys, before pretending to type their names on their paper keyboards.

In the next lesson, the children rotated around various typing-themed activities. A balloon-popping typing game proved very popular (**helloworld.cc/balloongame**), followed by children practising their typing skills using Word to type words or sentences, depending on their ability. They were given the flexibility to choose what they typed, with some just typing the names of family members or animals. Some even managed to start writing their own little stories before rotating to their next activity. The children also played keyboard bingo with their colouring sheets, drawing letters and placing counters on their keyboards with the aim of being the first one to get three in a row. One of our favourite activities, though, was fixing a giant's keyboard!

I displayed George the Giant on the screen, a character from the book *The Smartest Giant in Town*, with the rhyme: "George was a giant, the saddest giant in town. He wanted to type a letter, but his keyboard had broken down." In readiness for the activity, I had made a large vinyl picture of an empty keyboard and cut-outs of the missing keys. The children worked in groups, using their colouring sheets as a guide, to put all the keys back in the correct places. They really enjoyed this challenge and worked well together to fix the keyboard for George.

It's important to keep our subject engaging for young children and not lose sight of simple digital literacy skills. Children love being given the opportunity and freedom to type without constraints, so let's keep the magic of touch-typing alive and give them opportunities to just type! (HW)


Child using a colouring sheet as a guide to place the letters in the correct places on the 'giant's keyboard'

NICKI COOPER
Nicki is a computing curriculum specialist at U Can Too, supporting computing education through workshops and teacher training. She has spent the past year and a half volunteering in her local primary school, delivering computing activities for pupils from early years foundation stage up to age seven (**@GeekyNicki**).

IMPROVING DIVERSITY AND REPRESENTATION

Chris Lovell shares how primary-school teachers can help improve diversity and inclusion in the future STEM workforce

Research from the British Computer Society (BCS) has shown that women make up just 20 percent of the UK's IT industry workforce, and that only 11 percent of IT directors are from an ethnic minority background (helloworld.cc/BCSdiversity). Research from the Raspberry Pi Foundation (helloworld.cc/GBIC) has found that a variety of factors are responsible for women not seeing computing as a future career, including a feeling of not belonging to the subject or its community; a lack of sustained encouragement; and a lack of computing role models when they are learning the subject.

Digital Schoolhouse (DSH), an organisation offering free creative computing workshops to primary schools in the UK, has completed its own research examining how inclusive its programme is (**helloworld.cc/DSH2021**). In this article, I'm going to briefly explore a couple of the recommendations from its final report, focusing on the connection between representation in education and ensuring that we have a diverse and inclusive STEM workforce in the future.

Instilling a sense of belonging

One of the recommendations of this report is for lead teachers in the DSH programme to make use of DSH's diverse and inclusive One Minute Mentor (OMM) career resources (**helloworld.cc/OMM**). These include short videos that aim to inspire pupils with the breadth of roles available in creative digital industries. For example, in one of the videos, Ranjani, a producer from Robot Teddy, explains her role in ensuring that Ubisoft's video games are shipped on time. Ranjani advises viewers, particularly those from underrepresented backgrounds, to never self-disqualify from applying for a job, volunteering for a task, or going to an event. In another video, Jana, an art director for Yaldi Games, advises young viewers to build a portfolio that demonstrates their passion for the games industry. I have found that these videos give a real-life context to the topics we are studying in the classroom, and they show that careers in the digital industry are open to everyone. I use these videos as a starter activity in my lessons, and I have found that they prompt pupils to ask questions about the wider careers available in digital industries.

In my school, we similarly aim to encourage a sense of belonging in the subject by contributing to whole-school initiatives such as Black History Month. For example, primary-age pupils explored a variety of inspiring role models relevant to different subjects. In computing, pupils were encouraged to find out information about Jerry Lawson, a Black video games pioneer. Pupils shared with me that this initiative increased their awareness of how the voices and experiences of Black people have contributed to the development of technology that we use every day. This exposure to a diverse array of role models links to another important recommendation from the report: using representative imagery in resources. If pupils can see themselves in the images in the worksheets and the activities they use in their computing lessons, then they may be encouraged to feel that the subject is 'for them' and continue pursuing their computing studies.

Diverse role models and other forms of representation help ensure that all children see that they can aspire to all opportunities, regardless of their background, gender, or ethnicity. I believe that it is important to start these conversations early with children to enable them to develop their fondness for computing into a future career that they will love. (HW)

CHRIS LOVELL
Chris is the head of computing at Ashfold School, UK (**@ashfoldcomp**).

TEACHING COMPUTING AS A NON-SPECIALIST

We hear the ideas and advice of two teachers without a background in computer science who teach computing to primary-aged learners

JONATHAN TOWERS: BRECKON HILL PRIMARY SCHOOL, UK

How long have you been teaching?
I am currently in my second year of teaching primary after gaining my PGCE (England's teaching qualification) in July 2021. I have been teaching computing since qualifying.

What does a typical day of teaching look like for you?
Our school day always starts with independent reading and settling into the school routine. Our lessons include core subjects (English, maths, and reading) and foundation subjects (science, history, computing, etc.). We usually fit a one-hour computing lesson into each week, and it is one our students look forward to!

How do you approach teaching computing?
At our school we aim to make our computing lessons engaging for students by using devices such as laptops and iPads. Making use of data loggers and Crumbles also helps keep students engaged. In our current unit of photo editing, students use these devices to take and edit their own images to both gain an understanding of photo editing and allow them to take pride in their work.

What things worry you most in computing lessons, and how do you overcome this?
When teaching computing, my main worry is not being able to answer students' questions correctly. To combat this, I ensure I read up on subject knowledge required for that lesson beforehand and liaise with the computing subject lead at our school if I have any queries or concerns. Our subject lead regularly monitors lessons to ensure computing is being taught effectively across the school.

What excites you most about computing lessons?
I look forward to witnessing the light-bulb moment in lessons when students finally understand a difficult concept. This particularly occurs in computing when a topic is abstract or a process is not visible. For example, when I was teaching networks, we physically built a network using the students themselves. We used string to connect pupils who acted like computers, and other students were network switches who passed messages around the network. Not only was this active, but it also helped students understand how messages were passed between computers in a network. Seeing that moment of understanding was great!

How do you improve your own computing knowledge and understanding?
I like to play with the software that students will come across before using it in lessons. As a school, we are also currently receiving professional development to help us teach computing in an engaging and enjoyable way.

Do you have any tips for other primary teachers without a computing specialism?
My top tip would be to dive in and give it a go. It can be frustrating for students when things don't work the first time, so I ensure I try things beforehand so I can troubleshoot problems in class. I also encourage my students to keep trying, as resilience is a valuable life skill as well as being needed in computing lessons. This is advice that I give myself too!

PREETI WILLIAMS: CLAREMONT FAN COURT SCHOOL, UK

Why did you start teaching computing at primary?
Computing is not my background; I trained and worked as an EYFS (early years foundation stage) teacher. However, over the years, I have seen first-hand how technology in the classroom can impact learning and ignite that spark of curiosity and joy on a child's face. This is what drew me to the challenge of becoming a computing and Tech-Ed teacher, and what keeps me motivated when faced with new challenges.

What things worry you most in computing lessons, and how do you overcome this?
When I was first appointed, I remember feeling overwhelmed as I wasn't sure where to start. With the support of school leadership and guidance from my friend and mentor Emily Jones (Harrow International School, Bangkok), I audited and then rewrote our curriculum to create something more pupil-focused and engaging, with higher expectations and clearer learning outcomes. Never be afraid to ask for help!

How do you improve your own computing knowledge and understanding?
I take personal responsibility for being up to date with new initiatives and work hard to further develop my knowledge and skills. I am a member of the Computing at School community and other computing leaders' groups, where ideas are shared freely and people inspire one another. I attend webinars such as Apple Regional Training events, and have links with a local Apple Distinguished School and connections with former colleagues working across the international circuit. I also regularly bring work home to practise on my own children. Without the support of the community and my own commitment to learning new material, it would be impossible to teach students effectively. Every teacher, regardless of age range or specialism, will recognise this!

Do you have any tips for other primary teachers without a computing specialism?
I realise that it is easy to feel one step behind in the world of technology. But as my passion for learning and teaching the subject has grown, I feel more confident applying my knowledge and adapting ideas I have seen. My attitude is now, 'I don't know everything, but I can surely find out how to do anything' — a mantra I encourage readers to take up! Positive comments from colleagues, parents, and students have been an immense help, too, so remember these and hold onto them. This real-world feedback from those who matter most keeps me keen, engaged, and willing to put in the extra hours needed to succeed in a place I never imagined I would find myself! And I love every minute of it!

Getting to see pupils' light-bulb moments can be incredibly motivating

REBOOTING A COMPUTING DEPARTMENT

Catherine Archer offers a toolkit for new or established primary computing leads wanting to revitalise their computing curriculum on a limited budget

When faced with updating or creating a curriculum, the sheer volume of available resources can be overwhelming. Here is my tried and tested toolkit for getting up and running. (HW)

CATHERINE ARCHER
Catherine is an experienced primary computing head of subject and a secondary computer science head of department. She has worked in the UK and internationally for 15 years and is currently based at a school in Malaysia.

Don't reinvent the wheel	Free curriculum plans, lessons, and assessment documents are available online, so you don't have to start from scratch. Sure, they always contain aspects you won't teach, but they are an excellent starting point. Customise them by regularly adding your own notes and changes to create a tailored learning journey for your students. My favourites are from The Computing Curriculum (**helloworld.cc/tcc**) and Kapow (**kapowprimary.com**).
Go unplugged	Have some unplugged activities ready to go for each year group, such as those from Barefoot (**barefootcomputing.org**), CS Unplugged (**csunplugged.org**), and Bebras (**bebras.uk**). Unplugged computational thinking activities are memorable ways of getting students to think, learn, and collaborate, and they are also lifesavers if you have any technical issues.
Knowledge organisers	I recommend signing up for a free Kapow account (**kapowprimary.com**) and downloading their concise knowledge organisers. These can be shared with students at the beginning of each unit, making key learning and assessment transparent.
Physical computing	If you're lucky enough to have a budget, even a small one, buy a class set of micro:bits. They come with lots of tutorials and inspirational projects that can be linked with cross-curricular units of work (**helloworld.cc/microbitprojects**).
Do you know everything?	I think we can all identify some gaps in our skills and knowledge. You can find help by following in-app tutorials; watching YouTube tutorials; taking online courses from the Raspberry Pi Foundation (for example, **helloworld.cc/primarycourse**) or Teach Computing (**helloworld.cc/tccourses**); joining a local computing community (such as CAS in the UK, **helloworld.cc/CAScommunities**) or Facebook group; and by following teachers on Twitter.
Get networking	IT support staff are your greatest resource. I like to invite IT support into my lessons to share their working knowledge, such as how the school network works, with students. They can also occasionally source old computers for students to dismantle. For upper-primary teachers, I suggest connecting with computer science secondary-school teachers in your local area or via online communities, such as CAS in the UK (**helloworld.cc/CAScommunities**). Finding out what skills they would like to see coming up from primary is an easy way to improve student transition to secondary. You could even ask if they would like to visit your school and teach a lesson there.
Future-proofing	The world of technology is continuously evolving, and our students' futures may well look quite different by the time they leave formal education. Computing has many opportunities to promote the broader life skills they will need. For example, you could explicitly embed growth mindset attitudes (**helloworld.cc/growthmindset**) into the curriculum via year-group themes such as 'risk-takers' and 'critical thinkers', or turn tasks into open-ended/real-world challenges to promote attributes such as creativity, problem-solving, independence, and empathy.

PROGRAMMING TO GET YOU IN STITCHES

Margaret Low shares the value of programmatic embroidery, a cross-curricular activity linking maths, computing, and textiles

Physical computing is widely recognised as a useful and engaging way of introducing programming to children. However, I've found that the opportunities for creative activities sometimes take a back seat to technology. To combat this, we started using TurtleStitch (turtlestitch.org) in our outreach activities with local schools. This freely available software links maths, computing, design and technology, and textiles, and we've found that it supports a different kind of physical computing.

TurtleStitch is written in Snap! and interprets its pen module as a needle. Snap! is a block-based language and environment similar to Scratch. Children give instructions to a turtle, moving it around and turning it to create their desired pattern. The designs are then transformed into file formats for digital embroidery machines. Children can create designs that can be stitched within a few minutes, quickly bringing a tangible product, which can then be incorporated into other products, into the physical world.

Getting started

To support the use of TurtleStitch in the classroom, we've created free sets of resources (covering basic TurtleStitch skills and maths skills) and lesson plans (**helloworld.cc/warwickturtle**). We also regularly offer teacher CPD sessions (**helloworld.cc/turtlecpd**), which are open to anyone who is interested. For example, when Coventry became the UK City of Culture in 2021, we offered our Stitch in Time project to local primary schools, linking to Coventry's history of textiles and ribbon weaving (**helloworld.cc/stitchintime**). Children created patterns on a theme of the school's choosing, inspired by the city, and their work was exhibited at the University of Warwick's Resonate Festival in 2022.

The feedback has been very positive, with teachers reporting strong student engagement. They've found TurtleStitch supports the application of functional maths skills, such as geometry, ratios, and exploring shapes and coordinates. It gives students a reason to care about maths and how to apply it. TurtleStitch makes it easy to test, amend, and rerun code, encouraging children to explore code and check their understanding. They also get to experience the entire manufacturing process, going from their initial design, to writing the code, to seeing their design physically created.

I encourage you to explore the creative possibilities of using TurtleStitch in your classroom. It's quick to stitch patterns (compared to 3D printing, for example), and gets great engagement from young and old. Happy stitching! (HW)

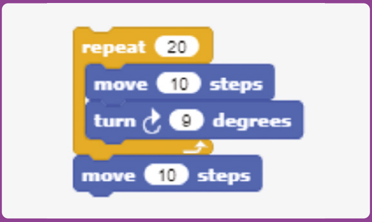
MARGARET LOW

Margaret is the director of outreach and widening participation at WMG, University of Warwick, UK. She has over 30 years of experience developing software for industry and teaching software development (**@megjlow**, **@megjlow@fosstodon.org**).

EXAMPLE PROJECT

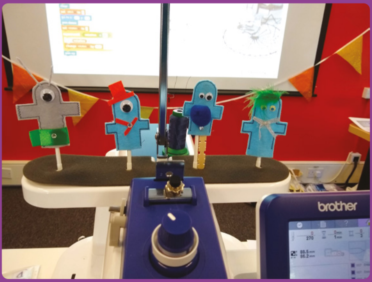
TurtleStitch project: create a critter
Aim: make a finger puppet
Materials: two pieces of felt

1. Using TurtleStitch, create an outline for a creature, making sure its body is wide enough to fit around a finger. Keep the pattern smaller than 10 cm x 10 cm.
2. Use the commands in **Figure 1** if you want to make a rounded head.
3. Stitch your design onto two layers of felt.
4. Cut around the stitches, adding hair or other features.

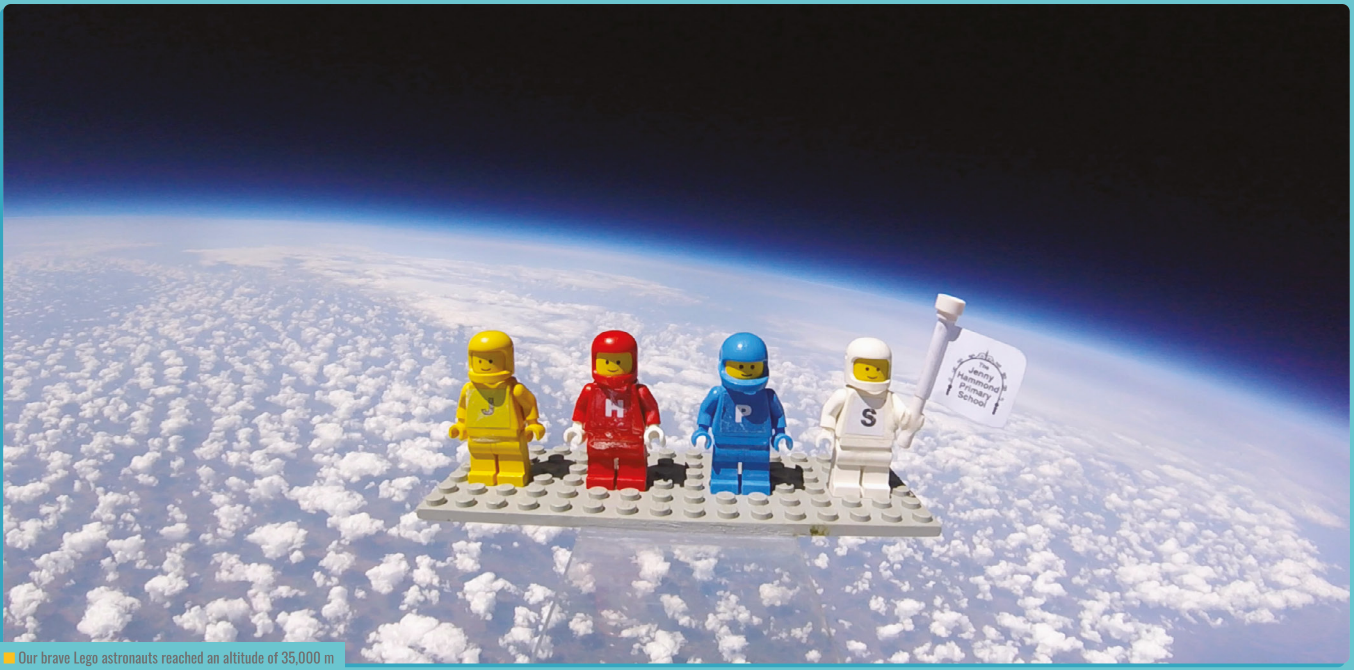


```
repeat 20
    move 10 steps
    turn ↻ 9 degrees
move 10 steps
```

Figure 1 Commands for a rounded head



Some of the work created by children in the 'create a critter' workshop



Our brave Lego astronauts reached an altitude of 35,000 m

SCHOOL, BUT NOT AS YOU KNOW IT

Peter Rutherford shares the value of large-scale, cross-curricular projects to keep students engaged in their learning as they approach the end of their primary-school years

At 4.30 am on a Thursday in July, I was standing in the playground of my school, surrounded by students and parents, some of whom were still in their pyjamas. The crowd had come to watch my Year 6 class (pupils aged ten to eleven) launch a high-altitude balloon carrying a payload containing a GoPro video camera, a Canon digital camera, a GPS tracker, and four brave Lego astronauts. The launch was the culmination of a project that had started two months earlier, the day after the children finished their SATs (formal exams in England).

Some 15 minutes later, we were given the go-ahead from air traffic control to launch the balloon before the airspace above London became too congested. Up until this point, I still had no idea whether it would actually work. As the crowd started the countdown, it dawned on me that this didn't matter now. The project was already a success. For the previous two months, our classroom was no longer a classroom but Mission Control. The children were not coming to school each day; they were going to work for the Jenny Hammond Space Agency. As the balloon rose into the atmosphere, the excited faces motivated me to plan more of these large-scale cross-curricular projects to actively engage pupils in their learning right up until the end of the academic year, and inspire and empower them for the future.

TOP TIPS

- Be led by the interests of the children
- Ask for help and support from experts
- Try to involve the whole school community
- Allow plenty of time
- Remember to take lots of photos and videos
- Don't be afraid to think big
- Don't worry if things go wrong

Engage

The summer term can be tough for a Year 6 teacher. The children have been working towards taking their SATs and, when they're over, keeping them engaged in their school work can be a challenge. A lack of engagement can then have a knock-on effect on behaviour, learning, and even attendance. When children are engaged in their work, these factors cease to be issues.

Last year, I planned a project based on retro video games, which you can read about on page 48 of Hello World issue 20 (helloworld.cc/20). We designed and built two arcade machines, and the children created their own games to play on them using the MakeCode platform. We kicked off the project by visiting a video arcade the day after pupils finished their SATs. They were hooked, and for the next couple of months, school felt less like school to them.

Projects like these can be equally engaging for the teachers. Doing something new each year keeps things fresh, and requires you to learn new skills before you can teach them.

Inspire

When I first announced the high-altitude balloon project, I was met with some scepticism from children, parents, and even colleagues about the feasibility of such an endeavour. It was a big idea, with so many hurdles to overcome and so many things that could go wrong. When we recovered the drone payload and the children could see the incredible images and video from the onboard cameras, though, I believe that they were truly inspired. They learnt that it's OK to have an ambitious idea that seems impossible at first. With enough hard work and plenty of mistakes along the way, they achieved something incredible. The sky was quite literally the limit.

Projects like this can also be a powerful way of inspiring pupils about their futures. During the arcade game project, for example, we had a presentation from an artist who works on creating characters for video games. This showed the children that it is possible to turn your interests and passions into a career. Although eleven-year-olds might seem young to be contemplating the world of work, there is definitely a benefit to opening children's eyes to different jobs and how the skills they are learning in school now could benefit them in the future.

The team of scientists and engineers at the Jenny Hammond Space Agency

Empower

I think one of the main benefits of these projects is that, from the students' point of view, it gives the curriculum a purpose. They suddenly understand the reason that they have been learning different skills across different subjects, as they now have a practical use for them. If they didn't know how to measure accurately or how to calculate averages, our arcade machines wouldn't have been the right size. If they hadn't calculated the correct amount of helium to put into the balloon, it wouldn't have risen to the desired altitude.

It is also wonderful to see how the projects have empowered pupils to continue the work in their own time, and the collaboration that's taking place between students. During the arcade game project, I had children who were not close friends arranging their own online meetings outside of the school day to continue working on their games together.

Working across the curriculum also allows children to develop their critical-thinking skills and helps them to develop a deeper understanding of the concepts they are learning.

Cross-curricular projects can also improve retention. By making connections between different subjects, students are more likely to remember what they have learnt and apply it in new contexts.

What's next?

I am currently planning a few projects for this academic year, but I'm undecided about which one will best suit my current cohort. The first idea is to create Strava Art (artworks made on a map using a GPS receiver to track physical activity). The children would draw pictures by planning routes around our local area. We would then head outdoors with GPS receivers to create their designs.

The second idea is a project to solve an issue the children have in school. We are lucky to have many lovely plants throughout our school buildings and grounds, but during the school holidays, one staff member has to come in to water them all regularly. The plan would be for pupils to build automated plant watering systems using microcontrollers, moisture sensors, and small water pumps.

The finished arcade machines painted by the winners of our design competition

The final idea is for the class to create and run a pop-up restaurant one evening, with their parents as the customers. They would have to devise a theme for the restaurant, build a website to take bookings, create a menu that they can produce quickly and at scale, and work together to ensure the smooth running of the evening. The potential for learning and collaboration is enormous. Watch this space! (HW)

PETER RUTHERFORD
Peter is the computing lead and Year 6 teacher at the Jenny Hammond Primary School in East London, UK.

ENGAGING GIRLS: CREATIVITY AND CONTEXT

Peter Marshman discusses the power of using a creative and purposeful context to improve female participation and performance in computer science

Why don't penguins' feet freeze, with temperatures around them getting as low as -70 degrees? We know their bodies stay warm with all that blubber beneath their skin, but how do they keep their webbed feet free of frostbite? This is an example of the hooks we've used at digit<all> (digitall.charity) to help encourage female participation and performance in computer science, with a view to developing girls' self-efficacy.

Mind the gap

The gender gap between male and female participation in computing is well known. The Roehampton Report 2019 identified that only one in five computer science students was female (**helloworld.cc/roehampton2019**). Predictably the post-education picture is no better: just 23 percent of STEM employees are female, and a worrying 5 percent of leadership positions in the technology sector are held by women (**helloworld.cc/womenintech**).

One method for helping reverse this trend is to consider the relevance and context of a topic when teaching it. Girls need to experience creativity, communication, and problem-solving within a society-relevant context (**helloworld.cc/arthur2022**). OK, so the state of penguins' feet may not be the most burning question the world is currently facing, but this is the sort of context that we've found engages and enthuses all types of students, and provides the 'why' that girls tend to need. It's also the sort of context that excites the seven-to-eleven age range this resource is made for. This age is vital because it's an early point at which girls start to decide what subjects are important and relevant to them.

Cold Feet

The free Cold Feet resource from digit<all> (**helloworld.cc/coldfeet**) sets up the scientific question of why penguins' feet don't freeze before introducing some Scratch code that helps develop this understanding, principally in the form of an ice-skating penguin being harassed by a llama. Students are taken through a series of coding activities that encourage them to interpret and modify existing code, with the aim of being able to use their penguin to draw some art as it ice skates.

We've aimed to produce a combination of context (the penguins), creativity (ice art), and prewritten code to help make the project accessible, appealing, and more likely to help girls realise their innate suitability for STEM subjects. The majority of our resources are either games with a real-world context, or are centred on making people's lives better, and both have enthused the girls taking part. Collaborative coding challenges, as well as the 'silly stuff' such as wearing bird masks, have also led to high participation rates and positive feedback. On average, the attendees at the events we've run using these resources have been 65 percent female.

So ... why don't penguins' feet freeze? If only I had the space to tell you! The answer can be found in our resource (**helloworld.cc/coldfeet**) — but maybe your time will be as well spent designing your own programme of study with a creative, purposeful context to engage young girls. Either way, it's exciting to think of the positive impact you'll be having on the next generation of female tech professionals. (HW)

TOP TIPS FOR ENGAGING GIRLS

- When designing resources or programmes of study, have a relevant, relatable context in mind
- Don't teach code; teach creative solutions that may require code
- Create a sense of belonging; choose topics that students can see themselves making important contributions to

PETER MARSHMAN
Peter is CEO and founder of the charity digit<all>, a secondary-sector expert for BCS, the Chartered Institute for IT, and the UK Code Week Ambassador (**@digitallcharity**).

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

OUR MOST POPULAR EPISODES

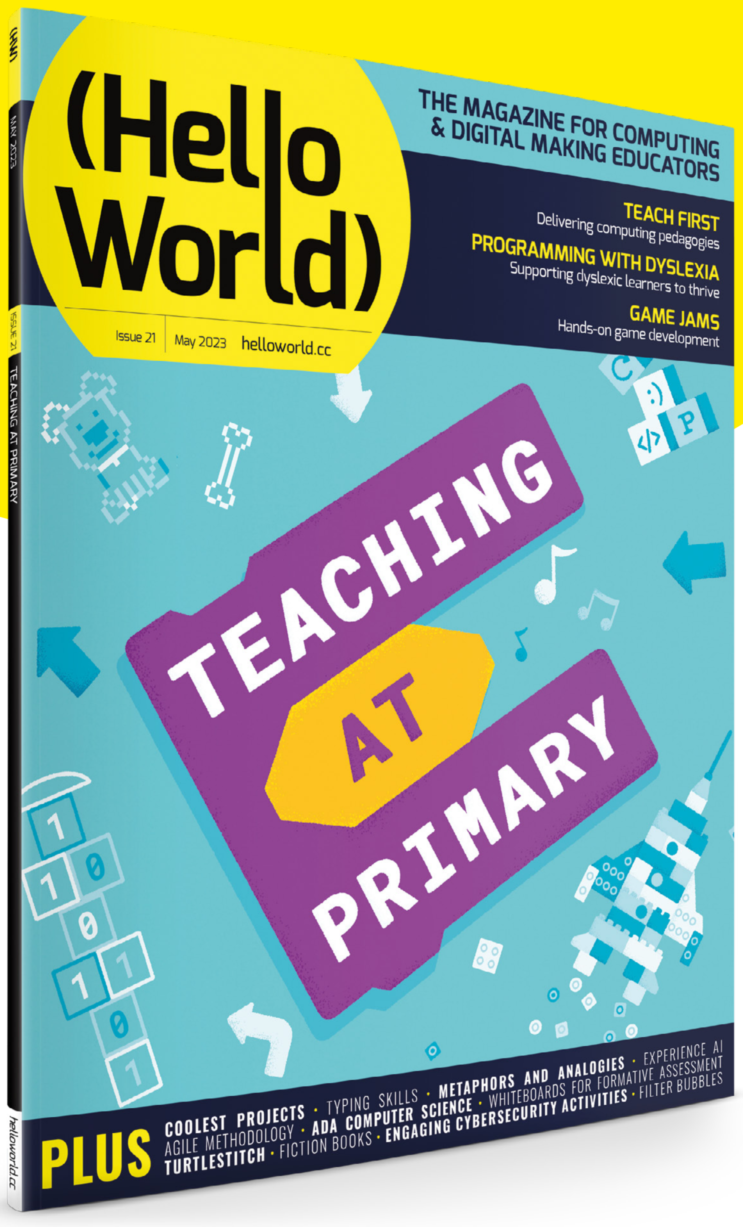
How moral is your machine? Ethics in computing education

Accessible and inclusive computing education: where to start?

How can we get everyone excited about code?

(HW)

SUBSCRIBE TODAY



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

TO SUBSCRIBE VISIT:

helloworld.cc/subscribe

(Hello World)
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
TEACH FIRST
Delivering computing pedagogies
PROGRAMMING WITH DYSLEXIA
Supporting dyslexic learners to thrive
GAME JAMS
Hands-on game development
TEACHING AT PRIMARY
PLUS

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

TEACH FIRST AND COMPUTING PEDAGOGIES

Daljit Shoker and **Rachel Arthur** share how they deliver computing pedagogies to their first-year computer science trainee teachers on the teacher training programme at Teach First

Teaching a class how to use new software? One of my trainee computing teachers has made great progress with her classes by 'modelling everything', a pedagogical principle that we have incorporated into our curriculum at Teach First. Teach First is a UK charity that trains top graduates to become teachers in schools serving disadvantaged communities. It aims to address educational inequality and ensure every child in the UK has access to an excellent education.

I (Daljit) work as the subject development lead on Teach First's computing training programme, preparing trainees to teach the national curriculum of England and Wales. We focus on developing participants' subject knowledge and leadership abilities, and place an emphasis on teaching in disadvantaged areas. The programme includes a combination of in-person and online delivery, in which we develop participants' teaching of computer science, information technology, and digital literacy skills using subject-specific pedagogical principles.

Daljit Shoker and Jonathan Usherwood, the subject development leads at Teach First, delivering an in-person computing seminar to their first-year trainee teachers

Twelve pedagogical principles

Writing the computing curriculum for Teach First was an exciting opportunity, particularly as I (Rachel) was starting from a blank slate. I wanted to create a curriculum that was not only accessible, practical, and engaging, but also one that helped trainees balance the amount of content they were learning. We only had a short amount of time with them before they went into the classroom, so we had to find a way to prepare them with effective strategies without overwhelming them.

To strike the right balance, I consulted Hello World's *The Big Book of Computing Pedagogy* (**helloworld.cc/bigbook**) and listed its twelve pedagogical principles (**helloworld.cc/12pps**). These principles act as guidelines for teaching computer science effectively. I then created a list of the topics that could be taught in computing and used the two lists to create a matrix of the pedagogies that teachers could use in different topics. It became apparent that some principles could be used in multiple lessons (for example, 'model everything'), while others were limited to more specific topics (for example, 'read and explore code first').

To make things easier for trainees, I prioritised the principles that were more applicable to multiple topics during the summer training, and then introduced others later in the course. This gave trainees a framework of the 'best bets' of how to teach the topics on their schemes of work, and we could focus observations around these principles. We make it clear to trainees that finding out the best principle for each concept is not an exact science. It may vary depending on the needs of their classes. We encourage them to experiment, but this has given my curriculum a clear and logical sequence, with a fantastic research base underpinning it. We've found that the bite-sized research articles for each principle in *The Big Book* provide quick and accessible advice and useful example content, which can help reduce planning time. Here are some examples I (Daljit) have shared with my trainee teachers.

Pedagogies in action

The research article about using the PRIMM (Predict–Run–Investigate–Modify–Make) pedagogy has supported a trainee teacher with structuring her programming lessons for students aged 14–16 (**helloworld.cc/bigbook** pages 22–24). The trainee noticed that the class were struggling to interpret Python code. They found reading and writing Python code extremely challenging, and weren't engaged as a result. During the trainee's subject development meeting, we (Daljit and the trainee) discussed how the PRIMM approach could be used to break down a program into steps. This would allow pupils to analyse and question the code they saw, and could therefore support their understanding. Since the trainee teacher introduced this approach when teaching a series of lessons, she has noticed an improvement in students' programming comprehension, as well as increased confidence when students began to write their own code.

When carrying out a lesson observation with another trainee teacher who was struggling with low-level disruption, I (Daljit) noticed that the teacher was spending a significant amount of time providing detailed explanations to the class. Students spent large chunks of the lesson listening to large volumes of complex information. This could potentially result in cognitive overload and could be contributing to the off-task behaviour being displayed during the lesson. To help reduce the cognitive load, I wanted to encourage students to think and recall their prior knowledge of a topic instead. I shared with the trainee a research article about concept maps that explains how to use these as a tool for teaching and assessing knowledge (**helloworld.cc/bigbook** pages 10–12).

During the trainee teacher's development review meeting, I modelled how they could use a concept map to capture students' prior knowledge of a topic. I then encouraged the trainee to use this approach to summarise key information. Students could then extend their concept maps when new knowledge was introduced, allowing connections to be made to existing schemas. I met with the trainee a few weeks later, and they found that using concept maps had encouraged pupils to recap and recall prior knowledge. Students were engaged in generative learning, summarising new learning using a different-coloured pen. This approach resulted in students having their own visual aids for revision, creating a sense of achievement and giving them the sense that they were making progress. The trainee teacher now sets the class homework to revise from their concept maps when preparing for upcoming assessments.

" THE APPROACH IMPROVED THE STUDENTS' UNDERSTANDING OF PROGRAMMING

The trainees have been asked to identify any issues or misconceptions in their computing classes when teaching at their employment schools. They are now writing assignments in which they collect data on one focus class of students, to measure the impact of implementing specific computing pedagogies into their classroom practice to address these misconceptions. As part of these assignments, they are carrying out literature reviews and critically reviewing research, including research from *The Big Book*, and this supports the trainees in becoming evidence-informed practitioners. We will be going into more depth about the findings of our trainee teachers in another article coming up in the next issue! (HW)

DALJIT SHOKER & RACHEL ARTHUR

Daljit is a subject development lead for computing at Teach First. She delivers the computing curriculum to trainee computer science teachers in the UK and supports them with developing their subject knowledge and teaching approaches. Previously, Daljit taught ICT/computing for 18 years (**@DaljitShoker**). Rachel is the head of computing, programme design, at Teach First. She has previously been a head of computing and assistant head teacher in schools across the UK (**@rarthurtweets**).

THE POWER OF PROJECTS TO SUPPORT LEARNING

Helen Gardner takes a look at how creating digital tech projects can boost learners' skills, resilience, and motivation

Coolest Projects is the Raspberry Pi Foundation's free worldwide showcase and celebration of all the amazing things young people create with technology. If you've never heard of it, it's a little bit like a science fair for computing. Young people share the digital makes they're proud of, receive personalised feedback from judges, and get inspired by all the other incredible digital creations their peers and friends make.

Coolest Projects events have been running online and in-person worldwide since 2012, and well over 10,000 young people have now taken part and shared their digital projects. As programme manager for Coolest Projects, it's my complete privilege to look at the projects young people share with us — some wacky, some weird, all wonderful. Here's what I've learnt so far about why creating projects — big or small — is great at supporting young learners to develop their skills.

Projects build resilience

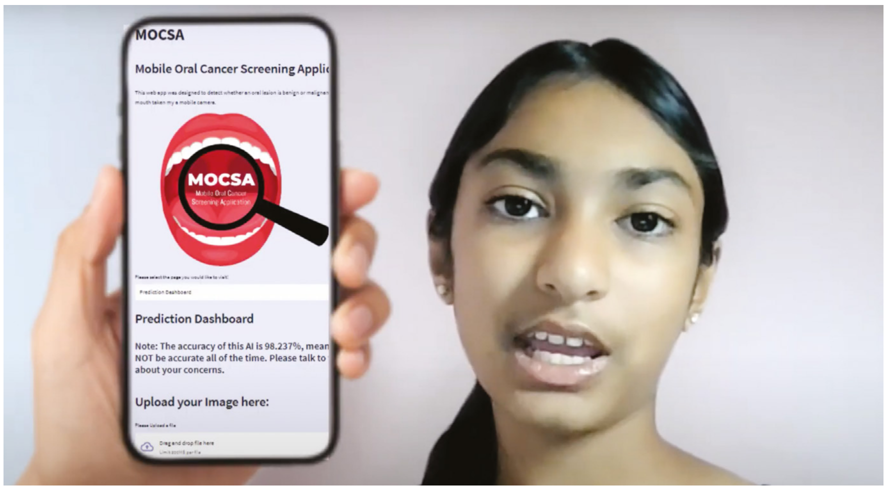
Resilience is an important skill for computing learners. Writing any code, for example, inevitably involves failures and setbacks, and entails a process of debugging and refining that is all part of learning. Working on one project and following this process of testing, debugging, and refining to complete it can really boost a learner's perseverance and resilience, especially if they have an opportunity to reflect on what they learnt.

Dafne (age twelve) from Turkey created the Treacherous Travel game in Scratch (**helloworld.cc/treacheroustravel**), after lots of hard work. "I wanted to make a parkour game for a while. In my attempts in the past, I failed. But this time, I got help and created a parkour game." Dafne's efforts to create the game helped her solve problems and think in different ways: "I was going to make this game with lots of levels. But difficulties I had made me spend lots of time on two levels, and instead of creating new levels, I decided to improve my two levels." Dafne reflects, "I work[ed] really hard to find the solution … It was actually really fun to test and make!"

Projects connect with what matters

Research has shown that context matters in motivating learners to use their skills, and that it matters for some more than others (see the article on page 10 for more about this). We know that while many boys become absorbed by the content in computing courses, for many girls, the context for using computing skills is more important. This context needs to relate to relevant scenarios in which computing can solve problems they care about.

Mahima (age 13) from the USA, for example, created a mobile oral cancer screening app (MOCSA) to tackle a problem she found close to home (**helloworld.cc/MOCSA**). She says, "When I visited our village in India last summer, I saw that

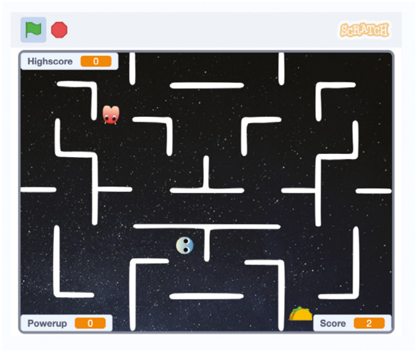

MOCSA
Mobile Oral Cancer Screening Applic
MOCSA
Prediction Dashboard
Note: The accuracy of this AI is 98.237%, mean NOT be accurate all of the time. Please talk to about your concerns.
Upload your Image here:

■ Mahima's project is a mobile oral cancer screening app many people were chewing tobacco, which is a risk factor for oral cancer. They were not aware of this and didn't have access to dental care either. I wanted to help people in these conditions by using my passion for coding to create MOCSA." Mahima has even thought about how she could make improvements to make the project even more useful in future: "I would create a version with more accessibility features [and] also make the app available in multiple languages."

Dhruv (age eleven) from India was also inspired to solve a problem in his local community with his automated solar-panel cleaning system (**helloworld.cc/solarpanelclean**). Dhruv knew about the benefits of solar panels to generate clean energy, but wondered why they weren't kept clean to make sure they worked as efficiently as possible. "I faced difficulty in doing research on why the solar panels were so dirty and not being cleaned. I worked it out by speaking to people in the neighbourhood who had solar panels installed on their rooftops." Dhruv found out that in his neighbourhood, solar panels are cleaned manually, which is a time-consuming and often risky process, as the panels are very high up. He created his automated system using code in Lego Mindstorms EV3, controlling a submersible water pump and a motor attached to a microfibre cloth to clean panels. Dhruv's next steps are to try out his project in the community: "If I had more time and resources, I would have tried to implement this prototype on some of the actual sites in my neighbourhood."

"REFINING CODE MAKES LEARNERS MORE RESILIENT

Dhruv was inspired to solve a problem in his local community with his automated solar-panel cleaning system



Keira's Scratch game allowed her to experience the various aspects of creating, designing, and coding a game

Projects consolidate learning

Once learners have mastered the basics of a new computing skill, creating a project is a great next step for them to apply and consolidate their new-found learning. For Paul McCarten, champion [volunteer] at CoderDojo Tramore in Ireland, this is the real benefit of creating projects. "For us, Coolest Projects is the answer to 'What do I do next?' After your ninja [learner] has consumed all the project resources you can give them and is looking for a slightly bigger challenge to use their new-found coding skills on, Coolest Projects is the answer."

Applying new skills in a fun and challenging project was key for Keira (age 17) from the Philippines. She was inspired to code a game because she wanted to "gain better insight into the fundamentals of coding". Her Scratch game Poko's Taco Spree uses variables to control the position of the sprites and to broadcast messages to communicate between threads (**helloworld.cc/pokotaco**). Keira says, "Creating this game allowed me to experience the various aspects and processes of creating, designing, and coding a game."

The Raspberry Pi Foundation's project pathways (**helloworld.cc/projectpaths**) are a great place to find ideas for projects like Keira's that allow learners to put their skills into practice. At the end of every project path, there's an 'Invent' project, in which learners can create a totally unique project that uses all the skills they have gained up to that point.

Digital technology projects of any kind — beginner, advanced, and everything in between — are a brilliant way to bring computing skills together, build your learners' resilience, and motivate them. Check out the projects by Dafne, Dhruv, Keira, Mahima, and many more amazing young creators in the 2023 Coolest Projects showcase gallery for inspiration (**helloworld.cc/coolest2023**) and then start creating — projects are the coolest! (HW)

HELEN GARDNER
Helen is the programme manager for Coolest Projects at the Raspberry Pi Foundation.

PROGRAMMING WITH DYSLEXIA

Tina Fountain and **Simon Carter** explore how dyslexia can be a strength when learning to program and give their tips for using that strength in teaching and learning

LEARNING TO PROGRAM AS A DYSLEXIC LEARNER

As a student in the classroom, my handwriting was illegible and my work was always incomplete. Note-taking from the board was impossible and I just could not keep up. I could understand the work and participate verbally, but I could never get my ideas down on paper. I still struggle with this today. I would work really hard on my written assignments, only to have them returned covered in red ink with comments on how I must try harder. When I was growing up, dyslexia was unheard of. By the time I was 13, I had simply stopped trying and left school at 16.

It wasn't until I entered the workforce that I began to thrive. Learning new skills came easily to me, my confidence grew, and I started a journey that led me back to education. Only then did I discover the term 'dyslexia', and suddenly every struggle I had experienced made sense.

I began my programming journey with HTML and CSS, and I was able to learn these with relative ease to the level I needed to teach them. Despite my initial confidence, learning to code beyond HTML and CSS was challenging. I could understand the basic principles and read other people's code, but I struggled with coding myself. Attending a Java course was one of the worst experiences of my life. The tutor expected us to copy her live coding as she went; it was like copying off the board at school, and I could not keep up. When I asked for help, she simply suggested that programming was not for me.

Luckily, I needed to learn to code for my job, so I dusted down my pride and searched for a way to teach myself. MIT's App Inventor (**appinventor.mit.edu**) was particularly helpful in building my skills and confidence, as it used a block-based development environment. Here, I was able to gain confidence and truly understand programming concepts. Learning text-based languages was then far more achievable. I am now confident in programming in multiple languages, and I finally conquered the basics of Java.

I found that writing out the syntax on index cards, along with simple applications of the concept, helped me immensely when learning Python. I needed time away from the screen to digest the requirements and to refer back to examples when tackling new problems. Planning on paper also helped me overcome the fear of the blank screen, and I now use this practice in the classroom with my students.

My dyslexia journey has been challenging, but it has also taught me the importance of understanding and accepting our unique learning styles. With perseverance and the right tools, anyone can learn to code.

TINA FOUNTAIN

Tina has over 20 years of teaching experience and currently teaches computing in a British school in Madrid. She is also part of the steering committee for the South East Asia Computer Science Teachers' Association (**@TinaFountain7**).

TOP TIPS TO SUPPORT DYSLEXIC LEARNERS

- Dyslexia can affect reading comprehension, which can make it difficult to understand complex instructions. Break down a task for students using bullet points rather than presenting paragraphed challenges.
- Dyslexia can also impact writing/typing skills, making writing and debugging code challenging. Provide a simple flowchart of debugging steps and lists of common error messages.
- Give students samples of code as a reference that they can use and adapt. Index cards are great for this purpose.
- Allow students time to annotate code and colour-coordinate variables or subroutine calls.
- Use videos for homework tasks. Videos will allow students to digest content in their own time and make notes without the pressure of keeping pace with the rest of the class.

ASSISTING DYSLEXIC PROGRAMMERS

Dyslexia is a condition that affects the incoming and outgoing signals by which our brain interacts with the world. It most commonly manifests itself as difficulties with reading, writing, processing auditory information, and performing arithmetic calculations.

We might expect that those with difficulties in reading and writing would have similar challenges in reading and writing computer programs. However, different and compensatory abilities make many dyslexic people exceptional programmers. Dyslexic programmers are helped by the fact that reading code is not like reading other forms of English. With only 33 keywords and 67 built-in functions in Python, for example, there are far fewer words to learn. Additionally, there is 100 percent consistency in the way keywords are used and spelled.

Dyslexic students also think visually, although not sequentially. They can perceive the various elements of a problem simultaneously, and so structure, hierarchy, and dependency can come more easily than to those who are not dyslexic. And therein lies probably the most powerful learning point I have come across in reading around this topic — that we should not lower our expectations of dyslexic programmers; we should merely expect different strengths. When they leave our classrooms, dyslexic programmers can become incredibly valuable systems analysts, members of software development teams, and software engineers. You can read more about why people with dyslexia make good programmers at **helloworld.cc/stein2018**.

The choice and set-up of the programming environment, the IDE, is a key factor to consider. In assessing a programming assignment for a particular dyslexic student, for example, I was struck by two things. On the one hand, his coded solutions worked as well as anyone else's. Keywords and instructions were spelled accurately throughout. He told me that his IDE's IntelliSense feature (in Visual Studio), which gave him instant feedback about whether a word was spelled correctly and colour-coded keywords, was really important in his mastery of Python. This, combined with the frequency of use of these keywords, enabled him to achieve a high level of accuracy.

On the other hand, other aspects of his code were harder to follow because he was using one- or two-letter identifiers throughout. We teach students that identifiers should be descriptive and meaningful. He confirmed that he prefers identifiers to be as short as possible to minimise the likelihood of spelling errors and inconsistencies. It made complete sense to him and, on reflection, to me. We needed to work on developing a consistent and recognisable system, maybe using comment lines early on to list identifiers and their meanings in a way that would not cause errors later on. His code could still be self-documenting, but in a different way.

In her blog article '*Dyslexia and Coding*', Joanna Leng draws attention to the need to spend time selecting and customising a coding environment (**helloworld.cc/leng2017**). This will include setting up fonts (specialist fonts such as OpenDyslexic and Lexie Readable, or more standard sans serif fonts such as Verdana, Tahoma, Arial, and Calibri), text size (12–14pt), spacing between letters and lines, and background colours/themes such as dark text on a light non-white background.

Enabling tab completion in your IDE can add your identifiers to drop-down autosuggestions. Sometimes, new modules will need to be imported. There are VS Code extensions for spell-checking and colour-coding pairs of brackets (**helloworld.cc/Vsdyslexia**). It is important to remember that there is no one-size-fits-all solution. But if teachers have chosen an IDE that allows customisation, a little time spent personalising it for dyslexic students can make a world of difference to their experience. (HW)

“ DIFFERENT AND COMPENSATORY ABILITIES MAKE MANY DYSLEXIC PEOPLE EXCEPTIONAL COMPUTER PROGRAMMERS

SIMON CARTER
Simon is head of computing at Hardenhuish School in Wiltshire, UK. He has been teaching computing for over 25 years and is a founder and steering committee member of the South East Asia Computer Science Teachers' Association.

Students work in teams, often beyond normal school hours, to make their games

GAME JAMS

Jared Rigby discusses the educational benefits of game jams to build bridges between computer science and other subjects

As a computer science educator who regularly gets to interact with students from primary school all the way through to the end of high school, there is a constant thread that exists, irrespective of age. When students are asked, 'Why do you want to study computer science?', one of the most common responses is, 'Because I want to make my own video games.' This intrinsic motivator allows students to directly map concepts they have learnt onto massively popular real-world products they have probably interacted with all their lives.

Making a video game, though, is a complex undertaking. Building something of value requires multiple hours of debugging, testing, and user feedback. When taught as a standard unit of enquiry within the classroom, it can lead to a frustrating experience. A student's game-making journey can be a slow, disjointed process, allowing for only minimal progress week by week, and often finds students building a similar, if not the same, game as their classmates. I've been through this process myself, and it left me wanting to provide an alternative experience for students who want to gain a more uninterrupted, hands-on experience in game development, in which they can drive their own learning outcomes. Enter the game jam.

What is a game jam?

A game jam is a hackathon for making games. Participants work in teams and are challenged to design and build a video game in a limited period. Most standard jams range from one weekend — as with staple game jams such as Ludum Dare and the Global Game Jam — to one month — like GitHub's annual Game Off or the challenging js13kGames competition. Here at Dulwich College Beijing (and across our wider family of schools), we decided on a weekend. Students opted into three days of uninterrupted time, from Friday morning, when they were taken out of regular lessons, through to a submission deadline of 12 noon on Sunday.

A key feature of a game jam is the theme, a vital piece of information that's kept secret from the students until the very start of the hackathon. Students are expected to incorporate the theme into their game and use it to help focus their initial brainstorming. A good theme should be open to interpretation and allow students to create a wide variety of experiences. The theme can also be leveraged to encourage students to consider wider-reaching issues across societies and how they can use games to inform their audience.

For our group of schools, we've decided to have the theme relate to one of the UN's Sustainable Development Goals (SDGs) each year. For this year, we chose to start with a theme of 'Be less wasteful', which directly ties in with SDG 12, "Ensure sustainable consumption and production patterns." Using this theme as a guide, our students created games ranging from the expected ocean clean-up games and forest preservation platformers to more out-of-left-field ideas, such as games focused around a sentient trash can or leading a bureaucratic visit to a factory to ensure a safe and healthy work environment.

Interdisciplinary opportunities

Another added benefit of game making is the opportunity for interdisciplinary learning and collaboration. Game jams are not only an opportunity for your tech-savvy students to dive head-first into programming challenges, but they also present the chance to build cross-curricular links with other departments in your school. A successful game jam team will require programmers, artists, musicians, narrative designers, and marketers. This opens opportunities for other subjects to explore the process of game making within their subject, from drawing pixel artwork in art or graphic design to creating sound effects in music or film or writing a story for their games in English. There is also the opportunity to create whole-school events in which different age groups can compete using age-appropriate tools.

" STUDENTS LOVED WORKING WITH THEIR FRIENDS

Primary students can explore game making in Scratch or MakeCode, where they can apply their knowledge of basic programming concepts to make an interactive experience. High-school students can apply some higher-level computing concepts, such as modular design and object-oriented programming. They can also apply concepts like vectors, trigonometry, and forces from their maths and science classes using industry tools like Unity or Unreal.

After completing their games, students submit their work for judging. This process could be done by teachers at your school who enjoy gaming, or you can reach out to professionals in the game industry to provide feedback on the students' games. In my experience, I've found game developers and game researchers at universities to be a very helpful bunch who are happy to try out some games! This also adds gravitas to the event, as the students want to impress the experts.

Having just completed this process for the first time here at Dulwich, we're already looking to make this an annual event that we can improve year after year. When collecting feedback from our students on what they enjoyed about the game jam, common themes included how the event was fast-paced and required them to learn how to collaborate effectively. They also enjoyed the chance to work on a project over a weekend with their friends. An area for improvement that students identified is that they want even more time to work on their games in the future. Here's hoping we can find more time in next year's calendar to keep the excitement for computer science high! (HW)

GAME JAM SCHEDULE

Day 1: plan and prototype

- 8.15–8.30am Morning registration
- 8.30–9am Welcome talk and theme announcement
- 9–10.30am Initial brainstorming and planning
- 10.30–11am Morning break
- 11am–12.45pm Prototyping and development time
- 12.45–1.30pm Lunch
- 1.30–3.30pm Prototyping and development time

Day 2: reflect and improve

- 8.15–08.30am Morning registration
- 8.30–9am Show what you've made so far and receive feedback
- 9–9.30am Update development plan based on feedback
- 9.30–10.30am Game development time
- 10.30–11am Morning break
- 11am–12.45pm Game development time
- 12.45–1.30pm Lunch
- 1.30–3.30pm Game development time

The school day ends at 3.30pm, but students could continue working on their games from home.
The final deadline was set as 12 noon the next day to give students time to fix any final bugs and upload a copy of their game online for the judges to play.

JARED RIGBY
Jared is a learning technology coach and computer science educator at Dulwich College Beijing, China. He also runs a small indie game studio called Gameful Developments. Jared can be found across most social platforms as **@jazibobs**.

IT'S LIKE ...

Ian Needham provides examples of his favourite metaphors, similes, and analogies for helping students master difficult threshold concepts

Using metaphors, similes, and analogies is a long-established tool for teaching difficult threshold concepts — the fundamental concepts of a subject which, once understood and mastered, become embedded in a person's knowledge. It is especially important when teaching computer science, which is littered with technical, often abstract, and complicated threshold concepts.

Before we explore examples of these, it may help to define what is meant by metaphors, similes, and analogies:

- A **metaphor** is a figure of speech that makes a comparison by using one thing to mean another, for example, 'heart of stone' or 'night owl'.
- A **simile** is a figure of speech that compares one thing to another thing, usually with the words 'like' or 'as', for example, 'My dog eats like a pig', or 'She is as brave as a lion.'
- An **analogy** is a comparison between one thing and another, usually for the purpose of clarification or explanation. An example is 'Life is like a box of chocolates — you never know what you're gonna get', from the movie *Forrest Gump*.

Now that's out of the way, we can think about why and how we can use these in computing classes. For our students, the understanding of threshold concepts is both transformative and irreversible (**helloworld.cc/meyer2003**); it fundamentally alters how students think about a topic, and importantly, once a student has grasped a threshold concept fully, they are unlikely to ever forget it.

Metaphors, similes, and analogies can help us avoid misunderstandings and misconceptions (**helloworld.cc/bigbook**, page 108), and can also help bring a concept to life, especially if the example is easily relatable to students. Extreme care must be taken to choose examples that do not compound confusion, and they should be used in a semantic wave structure, with the teacher repacking the concept back into technical language (**helloworld.cc/bigbook**,

COMPUTING CONCEPT	METAPHOR/SIMILE/ANALOGY
SYSTEM ARCHITECTURE	
Computer hardware	Parts of the human body: • CPU — the brain • The motherboard — the central nervous system • The case — the skeleton • The power supply — arteries and veins • RAM — short-term memory • Hard drive — long-term memory • Speaker — mouth • Microphone — ears • Monitor — face • Webcam — eyes
Fetch–decode–execute cycle	• Processing orders in a restaurant (fetching the order, decoding how to make it, making it)
Core	• Multiple ordering points in a fast-food restaurant (several orders can be processed at once, but customers must be served in order, so there is an overhead in communicating to the order points)
RAM	• Your dinner table — food is brought out for the meal, but when the meal is over, the table is wiped
Virtual memory	• If your desk at home cannot hold all your books at once, you need to put some back in the bookcase when you're not using them, and take them out again when you need them
ROM	• Instructions for what you do when you wake up, such as get dressed and get ready for the day
Cache memory	• Pencil/brush holder in art
CPU registers	• Shopping bags for a dedicated purpose, for example vegetables, meat, or frozen food

NETWORKING

Protocols	• The rules for addressing a member of the royal family • How to borrow a book from the library
Layers	• How traffic moves on a road (the road surface, the vehicles, the drivers, the traffic lights, and signs)
Router	• A traffic police officer on an island sending traffic to the best route
Firewall	• A door steward only allowing people in who conform to the rules, e.g. wearing a tie and smart shoes • A drug-detection dog, sniffing out suspect packets using a pattern
IP address	• A telephone number for a new mobile phone — you have one for each device, and it can change on each device
MAC address	• Your National Insurance number — it always stays the same
Peer-to-peer network	• A class picnic where everyone brings something different and shares with everyone; no one is in charge
Client server	• Lunchtime on a school trip — the teacher gives out the food and deals with requests from students
Hosting	• Having a party at a trampoline park — it is cheaper than buying lots of trampolines for your home, and the park knows how to manage health and safety
Star network	• A river system — small tributaries carrying a small amount of water from larger rivers downstream
VPN	• A covered truck on the motorway — you can't see what is inside the truck

PROGRAMMING

Algorithm	• A recipe for making scones; the step-by-step instructions are like an algorithm
Sequencing	• Getting dressed in the correct order — for example, underpants under your trousers (unless you are a superhero)
Selection	• Choosing which TV channel you want to watch — you can only watch one at once
Iteration	• A carousel, going around several times and repeating the same actions
Difference between count-controlled and condition-controlled	• Count-controlled is clapping four times when the teacher raises their hand; condition-controlled is clapping while the teacher has their hand up and only stopping once their hand is lowered
Variable	• A box or container holding one thing, with a label on the outside
Arrays	• Chocolates in a box — the chocolate box is the array, and the number of chocolates changes as you eat them

TYPES OF SOFTWARE

Application software	• What you do at school — the learning that happens in your lessons and what you produce
System software	• The school building, rules, and facilities, such as the sports hall and dining hall
Operating system	• Ways you can use school resources, such as getting a book from the library, getting a meal from the cafeteria, or registration in the morning
Utility software	• Cleaners and maintenance staff — they look after the security, safety, and cleanliness of the school in the same way utility software does for computers
Program translators	• On holiday, translating a menu at a restaurant into English: ◦ Compiler — all at once ◦ Interpreter — line by line

page 46). More importantly, success in understanding ever more complex threshold concepts relies on a firm understanding of foundational threshold concepts. For example, students will not be able to understand selection and the use of operators without first having a firm grasp on variables and data types.

I have compiled some examples that I have used over the years, and I hope they will help with your teaching. I am sure you have your favourites — and perhaps you can even get some of your more advanced students to come up with their own! (HW)

“ METAPHORS, SIMILES, AND ANALOGIES CAN HELP BRING A CONCEPT TO LIFE

IAN NEEDHAM
Ian is a senior lecturer at the Institute of Childhood and Education at Leeds Trinity University, UK, and the course tutor on the PGCE in Computer Science with ICT. He is also the secondary lead for the NCCE Computing Hub for North Yorkshire, Leeds, and Wakefield. He previously worked as a class teacher and head of department in several schools across Yorkshire.

© Bits and Splits/stock.adobe.com

WHAT CAN YOU TRUST?

Ben Hall explores why the stuff we read online really matters

Recently, there has been a series of controversies centred around the reliability and impartiality of online information. The news has been full of stories concerning what neutrality should look like and the reliability of information, with trust in politicians around the world being questioned. The stuff we consume online really matters; the outcomes of elections and referendums have been influenced by social media. There is evidence that the use of Twitter by the Vote Leave campaign was instrumental in the result of the UK's 2016 Brexit referendum, a vote that will have a profound impact for decades to come. On a more individual level, and perhaps more targeted at young people, influencers such as Andrew Tate have used social media platforms to bring misogyny and hate speech into the mainstream and into the social feeds of highly impressionable young people.

Has it always been like this, or has the proliferation of social media made it more obvious? Let's look at some statistics to see a snapshot of the problem:

- 62 percent of information on the internet is unreliable (**helloworld.cc/fakenewsstats**)
- One in three people is unaware that content might be false or biased (**helloworld.cc/falsecontent**)
- False news is 70 percent more likely to be retweeted than true news (**helloworld.cc/retweetstats**)
- 27 percent of children who claimed to be confident in spotting misinformation were unable to identify a fake social media profile in practice (**helloworld.cc/falsecontent**)

These statements suggest that a toxic combination of misinformation is being spread online, exacerbated by a general ignorance of the scale of the problem among both children and adults. For me, the last statement is perhaps the most worrying — that a significant proportion of people think they can identify false information more effectively than they actually can. All of this creates a fertile breeding ground for unreliable content.

To further complicate matters, AI technology has advanced to the point where a simple prompt can generate coherent articles, essays, and arguments. This raises so many questions. How reliable is content made by tools such as ChatGPT? These tools do not have any inherent ways of validating the accuracy of the content they create. Indeed, Noble Ackerson, a former Google developer expert for product strategy, found that ChatGPT thought he was dead! Given this, you may find it surprising that some media outlets are reporting that the UK Government is considering authorising the use of ChatGPT for GCSE and A-level students. All of these issues are complex, confusing, and rapidly evolving.

"SHOULD YOU ONLY FOLLOW ACCOUNTS THAT SUPPORT YOUR VIEWS, OR SHOULD YOU BE OPEN TO OPPOSING OPINIONS?"

Echo chambers

When beginning to explore these issues with students, it is helpful for them to know who is controlling what they see on their social media feeds. Many services are increasingly driven by algorithms, taking choice — or at least active choice — away from the user. TikTok, for example, works by showing content based on users' previous likes or posts. This model results in users seeing content that is filtered to suit their interests, even though they have not actively chosen any preferences, effectively making them passive editors of their own streams. Users are putting an enormous amount of trust in an algorithm they will never see or understand. There is also evidence that TikTok's policies towards responsible use vary in different countries (**helloworld.cc/tiktok1**), and that extreme political views are presented (**helloworld.cc/tiktok2**).

And it's not just TikTok. Platforms such as Twitter, Instagram, and WhatsApp are all widely used by children. A 2022 study by Ofcom in the UK found that TikTok, WhatsApp, Instagram, and Facebook were all accessed by over 40 percent of 3–17-year-olds (including 16 percent of 3–4-year-olds using TikTok), with use rising with age (**helloworld.cc/ofcom2022**). These platforms are all routinely used to spread misinformation, and often, this is not even done by humans — yes, the bots are here to stay! We therefore need to look for ways to help our learners navigate this complex online world. Have you ever considered how balanced your social media feeds are? Do you (like me) tend to follow accounts which support your own views? Is that a healthy way to consume content, or should you be more willing to listen to opposing opinions? Should we suggest the same to learners? You can take a look at Nicolai's article on the next page to explore this idea practically.

Traditionally, teachers advise learners to check three trustworthy sources before deciding whether information is reliable. In today's world of social media echo chambers and algorithm-generated content, is this still a sensible guideline? The key here is for children to be able to discern firstly what a reliable source is, and secondly to distinguish between facts and opinions that are presented as facts. Social media is probably the worst place to start checking facts, but it's also the environment with which many children will be most familiar.

Building awareness

I believe this problem needs a holistic approach that goes beyond computing curricula. It is crucial that learners have the skills to look at content critically, so that they can make their own judgements as to its reliability or balance (for example, try the lesson plans at **helloworld.cc/tcc1**, **helloworld.cc/tcc2**, and **helloworld.cc/tcc3**). Learners also need to understand that traditional sources of 'truth', such as newspapers, have their own agendas, and that their content is intentionally not independent, balanced, or unbiased. Even broadcasters that have previously been viewed as neutral, such as the BBC in the UK, are having their impartiality questioned from both sides of the political divide.

Beyond traditional or social media, there are ways to check reliability. There are some telltale signs for identifying AI-generated content. Such content is unlikely to include *speeling misteaks*, grammatical *error's*, or *tyPos* — so you can be very confident that a human has written this article! There are tools you can use to analyse passages of text to predict the likelihood they were produced by AI, such as OpenAI's AI Classifier (**helloworld.cc/AIclassifier**). There are also some relatively simple guidelines you can follow to use AI in a way that is more likely to produce reliable content. For example, an AI system is more likely to return reliable results if it can draw from a variety of sources (**helloworld.cc/deepak2023**). By teaching these skills to students, we can help them to understand the limitations and the potential of these technologies.

Above all, it is our responsibility to foster curiosity in our students so that their default approach to any content is not only 'Can I trust this?', but also 'What can I do to make that decision more informed?' (HW)

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

EXPLORING THE FILTER BUBBLE

Nicolai Pöhner shares how students can learn about data mining by looking at filter bubbles in social networks

The latest changes to our computer science curriculum in Bavaria introduced the topic of data mining to our classrooms for the first time. These changes have expanded the topics of relational databases and data security in ninth grade (students aged 14–15) and serve as a first glance at artificial intelligence, another matter that is new to the curriculum for older pupils. So far, I have found that using a fake social network called InstaHub (a clone of Instagram, instahub.org) to teach databases has been quite popular. So, I thought why not continue this trend and teach data mining through the context of social networks too? This article gives some insights into a lesson sequence that I have tried.

Filter bubbles

Social networks usually aim to keep users on the platform for as long as possible. One way that platforms do this is by suggesting content that aligns with a user's interests. This leads to 'filter bubbles', within which users receive personalised feeds. Of course, this can work to the user's advantage, but filter bubbles also have an adverse impact. In political discussions, for example, you only get to see the content and views of like-minded people who do not share different perspectives on a topic. Constantly reinforcing your own standpoints in this way can eventually lead to more radical reactions when you do hear opposing views. Data mining is what is taking place to create these filter bubbles and is a relatable and motivating context for students, as they have probably all experienced its effects.

Unplugged activities are a good starting point to help students understand how filter bubbles are created. For example, I've found that the Bebras task in the boxout helps students recap their knowledge of graph data structure (a collection of nodes containing data, and edges representing a relationship between nodes). In the context of social networks, nodes represent the members of a social network and edges describe whether one member follows or is friends with another member. Students can then move on to looking at how the strength of a friendship in a social network can be measured. I teach students about the Jaccard index as one possible measure. Mathematically, this index compares members of two data sets, such as the interests of two people, and calculates how many are shared. The higher the percentage, the more interests those two people have in common. For example, two friends, A and B, have these interests:

BEBRAS TASK: SUPERSTAR

The social network TeeniGram allows its members to follow other members. Moreover, there are member groups in TeeniGram. Within a group, a member is called a 'superstar' if:

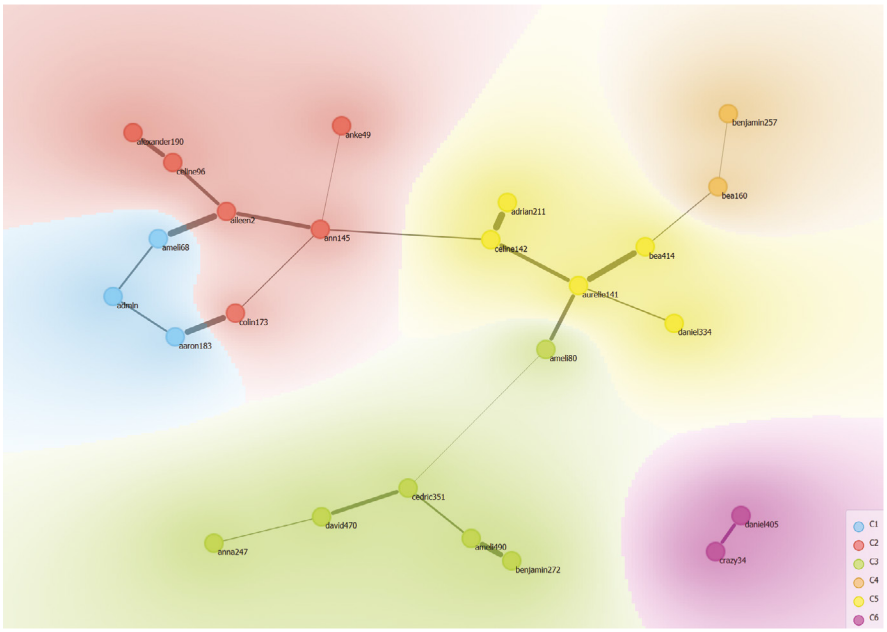
- They don't follow any other members of the group
- All other members follow them

For example, a TeeniGram group consists of the members Hailey, Selena, and Justin. Hailey follows Justin and Selena. Selena follows Justin. Justin doesn't follow any of the other members. Justin is the superstar of the group.

Another group consists of the following members: Alan, Don, Frances, Grace, and Robin.

- Alan follows Don and Grace
- Frances follows Alan, Grace, and Robin
- Don follows Grace and Robin
- Robin follows Alan and Grace

Who is the superstar of the group? (Answer: Grace)



Result of a cluster analysis conducted with Orange3, showing members of a social network and their relationships; groups or clusters of members are highlighted in colour

THE JACCARD INDEX

To teach students about the Jaccard index, randomly assign students five interests or let them choose their interests themselves. Then ask them to walk around the classroom, compare their interests with those of their classmates, and calculate how similar they are.

> TEACHING DATA MINING IN THE CONTEXT OF SOCIAL NETWORKS LETS US VIEW IT FROM A SOCIOCULTURAL PERSPECTIVE

- **Friend A:** {football, guitar, computers, science fiction novels, board games}
- **Friend B:** {piano, football, board games, computers, dogs}

We work out the percentage of shared interests by calculating the Jaccard index:

|{football, computers, board games}| / |{football, guitar, computers, science fiction novels, board games, piano, dogs}| = 3 / 7 = 0.429. So, the two friends A and B share ~42 percent of their interests. The second boxout offers an example activity so students can explore the Jaccard index.

Data mining in practice

Of course, real data sets are much bigger and include many more facets of data, so it's important to now move on to using a tool that helps students put data mining into practice. Orange3 (**orangedatamining.com**) is a powerful tool for data visualisation and analysis that allows even beginners to conduct their analyses through a drag-and-drop interface. You can use it in class to analyse big data sets of social networks using the Orange3 add-on for network analysis (**helloworld.cc/orangenetwork**). Students use different widgets to import their files, visualise the networks, and find clusters. The network files are simple Excel files that save pairs of friends within a social network, along with the strength of their relationship. You can use the Excel function RANDBETWEEN to generate a random number for the strength value and then compare the resulting clusters in your network. Make sure to check out the online material and try out the Orange3 example projects (**helloworld.cc/orangesocial**).

Teaching data mining in the context of social networks allows us to view it from a technological, sociocultural, and user-oriented perspective. The user-oriented perspective describes what students can actually do to take control and is a great way to wrap up a lesson sequence about data mining. You can encourage students to think about how they can take control by discussing how they can adjust their settings in social media apps, for example, by turning off account suggestions in Instagram to minimise the effect of filter bubbles.

All in all, the context of filter bubbles in social networks can help motivate your students, as they have likely all experienced their effects first-hand. Use their own experiences as a starting point and try the lesson sequence out for yourself. Bringing in the personal experiences of my students confirmed to me the importance of teaching them about data mining in social networks. Many of them were unaware of, or even disregarded, the negative consequences of their often poorly configured privacy settings in favour of the convenient features of their social media apps. (HW)

NICOLAI PÖHNER

Nicolai is a computer science teacher in Germany. He received his PhD in computer science education from the University of Würzburg, Germany, where he examined the effects of educational robotics competitions on students' learning (**@NPohner**).

SPRINTING TO SUCCESS

Victoria Berkowitz outlines how she incorporated an agile framework into her robotics class, and shares the benefits it has had for teaching and learning

We often teach content to prepare kids for the corporate world, but we should also consider what the corporate world can teach our kids. I recently incorporated agile methodology into my robotics course by introducing sprint boards, daily scrums, retrospectives, and team planning. This is an approach to working on projects incrementally and efficiently, and in my previous experience as a software developer, we would use agile to help us plan and reflect while developing a product.

I had been teaching robotics for a year before I realised that agile could solve some of the problems we were having as a class. I noticed that a lot of my kids might know the direction they wanted to go with their robots, but struggled to figure out the steps that were needed to get there. This would cause a lot of stalled periods for some project team members who did not know what to do when they were stumped, and also for those who did not have ownership of a task while something else was being done. My students struggled to communicate their needs and actions to each other, and it took a lot of prompting on my end to encourage the teams to have those discussions.

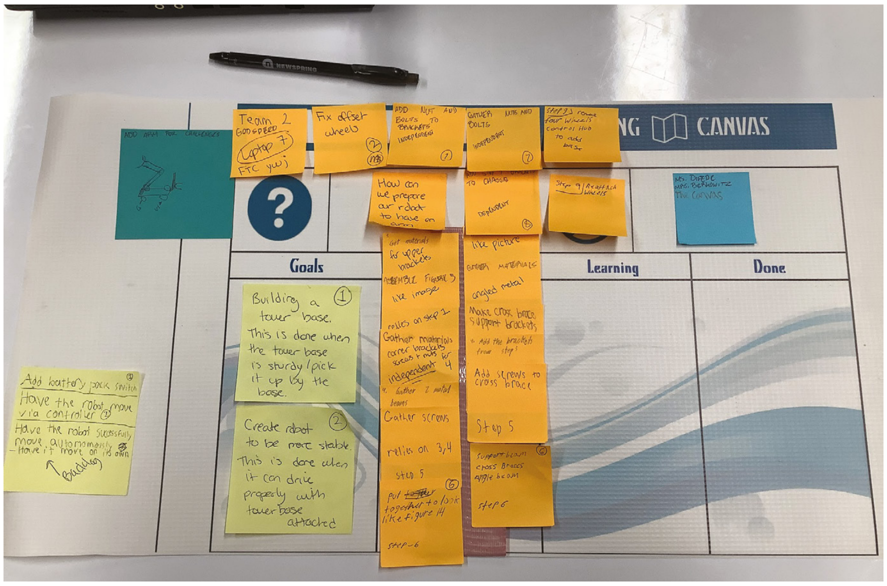
Incorporating agile

I started off by dividing my class into teams of two to four, before introducing them to sprint boards (**Figure 1**). Sprint boards are a visual tool used to help teams plan what they would like to accomplish during an upcoming 'sprint', or a defined period of time (a week in our case) in which planning, working, and reflecting take place. You can find free sprint board templates at **helloworld.cc/agileclassrooms**.

The sprints start off with team planning, requiring members to assess where they are with the project and what new goals they can set. This is where learners discuss what they want to accomplish and how that fits into the bigger picture of their goals. Some may decide to change goals based on the previous sprint, or to better meet their needs after reflecting. Students then put their goals onto a sticky note and place it on the goals section of the sprint board. Each goal is then decomposed into specific tasks, written onto sticky notes, and put into the tasks column.

Once sprint planning is complete, the teams get to work. They then move sticky-note goals from 'planned' to 'in progress/learning', to 'done', according to their progress. During the sprint period, daily scrum meetings take place. These are brief status meetings at which each team member shares what they worked on in the last class, what they plan to work on in this class, and whether they anticipate any setbacks or stumps. To support this, I created sentence starters with some variations for them to choose from, for example, 'Yesterday, I worked on ... ', 'Today I plan to work on ... ', and 'I might struggle with ... today because ... .' In addition to this, I would join their scrum meetings and model how I would share.

At the end of a sprint, teams hold a retrospective (**Figure 2**). This is supposed to be reflective in nature, and allow learners to share one thing they are proud of, one thing they can improve on, and possibly another thing they can continue to do. There are many variations of retrospectives, so I would switch them up at the end of each sprint to get my learners to reflect in different ways. For one retrospective, I prompted teams with 'Glow, grow, want to know', while for another, it was 'Appreciation, big "aha", apply to work life, and rate how well we did.'


Goals
Learning
Done
CANVAS
Team 2
Fix offset wheels
Building a tower base. This is done when the tower base is sturdy/pick it up by the base.
Create robot to be more stable. This is done when it can drive properly with tower base attached
Add battery pack switch
Have the robot move via controller
Have the robot successfully move autonomously - move it more on its own
How can we prepare our robot to have on
Add screws to cross brace
Make cross brace support brackets
Gather screws
Step 5
step 6
Like picture

■ **Figure 1** Sprint boards are a useful visual planning tool

Outcomes

My learners have gained so much from using agile in the classroom. Firstly, they've become a lot better at planning and achieving goals that fit into the bigger picture. Early on in their agile journey, my

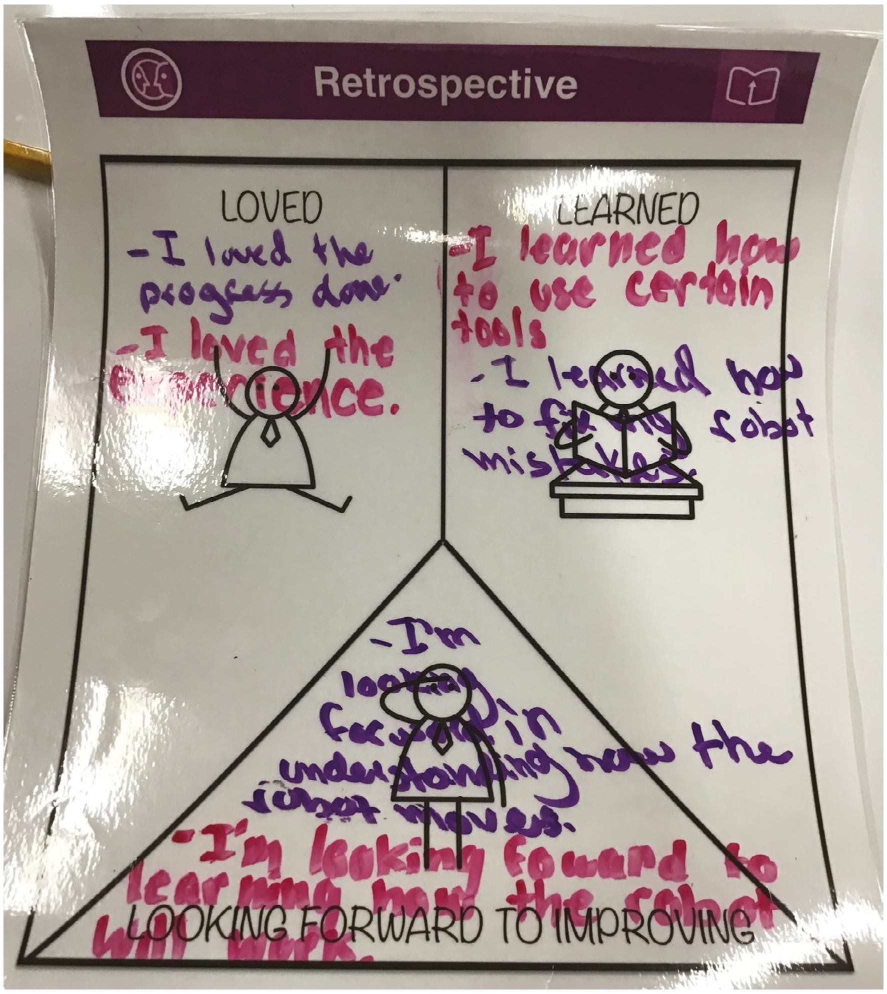

Retrospective
LOVED
-I loved the progress done.
-I loved the experience.
LEARNED
-I learned how to use certain tools
-I learned how to fix the robot mistakes.
LOOKING FORWARD TO IMPROVING
-I'm looking forward in understanding how the robot moves.
-I'm looking foward to learning how the robot will work.

Figure 2 Retrospectives offer an opportunity for students to reflect

VICTORIA BERKOWITZ

Victoria is a software developer turned computer science teacher. She works at Mineola High School in the USA, where she teaches GameMaker, Robotics, and AP CS Principles. She loves all things teaching, whether it be in public schools, volunteering, or children's ministry (**@v_berkowitz**).

learners created tasks that were often too vague or too big. For example, one team came up with the task of 'adding wheels to make the robot move'. But how many people would work on this task? How many wheels would be added? Where would the wheels be added? How does one team member know this task is complete? These were just some of the questions that needed answers in order for the teams to work effectively together. To address this problem, I introduced the idea of SMART (Specific, Measurable, Achievable, Relevant, and Time-Bound) goals, which has helped them create more meaningful tasks.

As a teacher, I've also found that agile has provided more transparency on the learning being done; I could easily walk past a group and glance at sprint boards to see what was in progress, along with what has been completed and by whom. Before implementing agile, I would have to probe a lot and make assumptions in order to assess productivity. Using agile also gave me a lot more time to converse with kids and build relationships with them while a sprint was taking place.

Another benefit of this approach was the efficiency gains over the previous year, when we were not using agile. Because of the sprint boards, someone could easily come to class late or return from an absence and still have multiple tasks to choose from. Teams this year progressed a lot faster with their robots than my teams last year, and I attribute this to the sprint board providing smoother collaboration and a platform for student voice. Because learners were able to choose their goals and then select and create their tasks, I noticed more buy-in to the work.

I also recognised that my learners became more able to reflect and use peer feedback to continuously improve. When we held retrospectives, team members provided useful feedback and ideas on how to improve the next sprint. One of my favourite things that came from using sprint boards was that my learners became able to see exactly what they had learnt and accomplished. This opened up so much discussion on how they could challenge themselves next time and how proud they were for meeting the goals they had set. I think this provided them with great confidence and motivation to do more, especially if they had surpassed goals they had set for themselves.

" SPRINT BOARDS BOOSTED LEARNERS' CONFIDENCE

Overall, I would encourage any group or project-oriented class to try using agile. While my robotics course is project-based, I have read about other teachers using agile for traditional courses in which tasks are co-created between teachers and learners. I suggest not worrying about sticking to every rule or aspect of agile, but moulding it to fit your classroom and its needs. (HW)

Youth culture, interests, and home and community factors all influence learners' beliefs

AREAS OF OPPORTUNITY

How can we start to tackle the underrepresentation of girls, Black, and ethnic minority learners studying computing?

Around the world, young people can increasingly learn about computing in classrooms and out-of-school clubs. Despite this, the statistics on the number of females, Black students, and those from ethnic minority groups taking computing exams or working in IT are dreadful. In the UK in 2020, for example, only 20 percent of IT specialists were female (helloworld.cc/BCS2021) and in 2019, just 3 percent of the tech workforce in the UK identified as Black (helloworld.cc/hired2019). A lack of diversity in computing is not a new problem, and despite many initiatives to address it, this dire situation remains.

Culturally relevant education theories

Since the 1990s, education theories have been developed to address inequity in teaching and learning, which centre around culture, such as culturally relevant pedagogy and culturally responsive teaching (**helloworld.cc/billings1995**). Recently, a computing-specific theory has been developed called culturally responsive computing (CRC) (**helloworld.cc/scott2014**). CRC states that digital and technological innovation is achievable for all students. Educators create experiences in which students learn about biases in technology development, reflect on their identities and cultures, and learn how to solve problems that are important to themselves and their communities by applying technology in innovative ways. When using these approaches, the aim is to increase engagement in computing and associated careers (**helloworld.cc/madkins2020**).

These theories were mostly developed and researched in the US, but what might they look like in other cultural contexts, and how might they be adapted for classrooms in other countries? To address this question, we at the Raspberry Pi Computing Education Research Centre have been researching this area with classroom teachers in England for several years now. To start our research, we reviewed existing research on the area (**helloworld.cc/leonard2021**) and worked with teachers and expert researchers to create culturally responsive computing guidelines (**helloworld.cc/CRguidelines**). These guidelines are organised into three sections: the curriculum, teaching approaches, and learning materials (**Figure 1**). Building on these guidelines, we then developed a set of ten areas of opportunity (AOs), and a checklist of prompts to help teachers review and adapt their classroom activities and resources (**Table 1**). We have thus far conducted two research studies relating to these AOs. Initially, we drew up the AOs to facilitate open discussions with teachers who were working with us on a research project funded by Google. We subsequently conducted a study, funded by the Cognizant Foundation, to evaluate their application in primary-classroom resource development. In the rest of this article, I will share a short introduction to the AOs. As you read about them, perhaps think about how they might apply in your teaching context.

“BY ADAPTING RESOURCES TO BE MORE CULTURALLY RELEVANT, WE CAN MAKE COMPUTING A MORE WELCOMING PLACE

Learners and teachers

The first two AOs concern finding out about our learners and ourselves. Many aspects of our lives influence our beliefs about computing. Take a moment and ask yourself, 'Do I belong in computer science, and in what capacity? What has led me to think this?' For young people, many factors will influence their beliefs, including youth culture, interests, and home and community factors (including heritage culture, religion, and political and economic factors, as well as the jobs and opinions of their peers and family). How a student learns is amplified or filtered by their values and beliefs, and this directly impacts motivation and outcomes (**helloworld.cc/newsome2015**).

Finding out about our learners' lives and how they influence their attitudes towards computing is therefore essential if we are to help them get the most out of learning opportunities. For very young learners, teachers may discuss pupils' lives with their families when they first join a school, or later through questionnaires, parent–teacher meetings, homework activities, and daily discussions. We found in our research that gathering data about the cultural context of students' lives was more challenging for secondary-school teachers. However, teachers intuitively made decisions about lesson plans and activities based on their awareness of their learners and their needs.

Just as beliefs amplify and filter how students learn, they also influence teachers, what they teach and how. Therefore, teachers need to reflect on their own views of computing and how they might change their practice. We found that schools sometimes run training courses on unconscious bias or about equity and diversity, but this learning was only sometimes used to review policies and lesson planning at the subject level.

Content and context

What is taught (the content) and the subject areas and topics in which it is taught (the context) form the basis of AOs three and four, and can clearly be related to culturally relevant teaching. In computing, some curricula focus the content on learning technical aspects such as programming. There are ample opportunities to build upon this with the social and ethical issues of technology, such as data bias in machine learning applications, or designing user interfaces to applications that are accessible for all.

The context in which computing activities are situated can also be used. For example, a primary-school teacher might teach programming by asking students to develop a quiz about environmental issues. Other examples are the instances and analogies teachers use to exemplify learning and make it more relevant, such as giving a list of careers that might be associated with programming. Recent research indicates that everyday-life contexts positively impact youth interest and engagement in computing (**helloworld.cc/ryoo2019**) and that relevant role model examples positively influence students' view of STEM to dispel myths and increase a sense of belonging (**helloworld.cc/gonzalez2020**). How teachers decide on the contexts and examples that are most familiar, relevant, memorable, and increase a sense of belonging for their cohort of students is likely to involve a complex

>

AREA OF OPPORTUNITY	DESCRIPTION
1. Learners	Find out about learners in order to reveal opportunities to adapt our teaching
2. Teachers	Find out about ourselves as practitioners to reflect on our cultural lens
3. Content	Review what is taught in terms of the content; add in extra culturally relevant content (e.g. about social justice/ethics, or data bias)
4. Context	Review contexts and examples used, to make teaching relevant and meaningful, and to contextualise and make connections
5. Accessibility	Make the content accessible and relevant for all learners
6. Activity	Provide opportunities for learners to participate in open-ended, inquiry-led, or problem-solving activities
7. Collaboration	Develop student-oriented learning through collaboration and structured group discussion
8. Student agency	Develop student-oriented learning through student choice
9. Materials	Review the learning environment (including learning materials) to increase accessibility and a sense of belonging, and to promote respect
10. Policy	Review related policies, processes, and training in your school and department

Table 1 The ten areas of opportunity for reviewing, creating, and adapting computing lessons

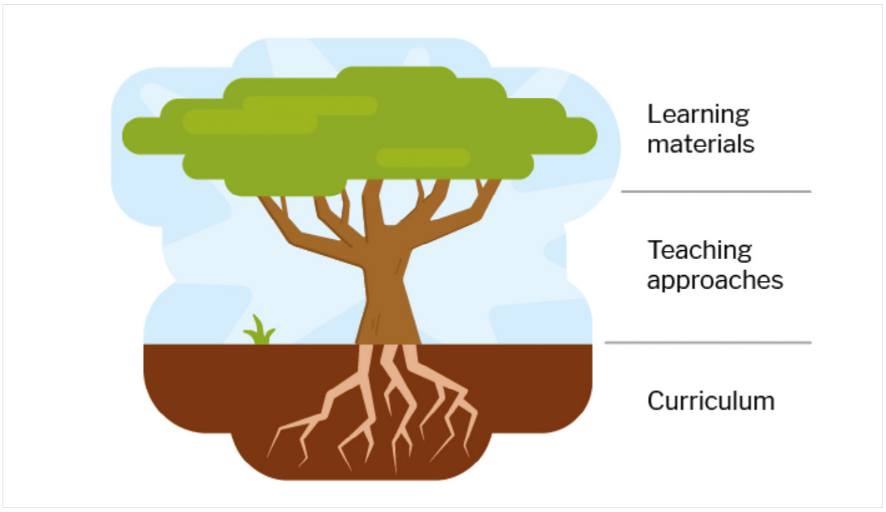


Figure 1 The guidelines are set out in three sections, which we have likened to the structure of a tree

decision-making process that requires more research.

Accessibility, activity, collaboration, and student agency

AOs five to eight particularly relate to **how** we teach computing. The fifth AO, accessibility, for example, asks teachers to reflect on their use of appropriate and accessible pedagogical approaches to ensure that all students can access learning. To help teachers make programming more accessible for all students, for example, the PRIMM lesson structure has been adopted by some teachers to carefully guide the process of learning to program (see pages 22–24 of *The Big Book of Computing Pedagogy,* **helloworld.cc/bigbook**, for more about PRIMM). The sixth AO, activity, hones in on open-ended and problem-solving activities to support students in thinking about different ways of solving problems.

Our seventh AO relates to student collaboration. Research studies indicate that peer instruction and pair programming (see pages 56–59 of **helloworld.cc/bigbook**), flipped classes (**helloworld.cc/latulipe2018**), and peer mentoring (**helloworld.cc/kulkarni2018**) are promising teaching approaches, as they can improve student outcomes, particularly for females (**helloworld.cc/morrison2021**). Currently, much of this research is at an undergraduate level. The eighth AO asks educators to help all students feel more engaged in computing lessons by offering them choices in terms of the content or context, or in terms of the way they might work. In computing lessons, when given the opportunity, students have been found to make different choices according to their interests (**helloworld.cc/spieler2018**) and to personalise their learning (**helloworld.cc/morales2019**) and so express themselves (**helloworld.cc/madkins2020**). However, some research has suggested that too much choice can overload some students and compromise outcomes (**helloworld.cc/beymer2015**). Teachers have to balance the increased motivation arising from choice, particularly for harder-to-reach students, against this overload.

> TOGETHER WE CAN MAKE COMPUTING MORE WELCOMING

Materials and policy

Often, learning materials represent students in terms of images and terminology (AO nine). Resources, and the physical class environment, enable students to see themselves and their communities being represented within computing (or not) (**helloworld.cc/leonard2021**). However, finding resources that do not reinforce stereotypes, and artefacts for new culturally responsive activities, can be difficult and time-consuming for teachers. The tenth and final AO is at the school level, and involves reviewing related policies and processes in the educational context. The task of introducing culturally relevant practices into computing classrooms is made more difficult without school policies and departmental or year group processes to support teachers' efforts.

We have suggested this set of AOs to support classroom teachers in England in reviewing their computing teaching activities with respect to culturally relevant pedagogy. The AOs are being used as a starting point in our research; we are still analysing data from some of our research and are midway through other studies, but we will share our findings as soon as we can. In the meantime, why not think through the ten areas for a lesson or unit of work that you teach? There is no doubt that teachers need to balance the areas of opportunity, thinking about how groups of students and individuals can be influenced by making changes in each area. One thing we have seen over and over again in our research is that teachers are very good at this; they think deeply about their students, and reflect on their own views as well as what and how they teach.

We hope this set of prompts will give practitioners a common language to discuss how we review and adapt resources to be more culturally relevant, and so together we can make computing a more welcoming, diverse, and inclusive place to be. (HW)

JANE WAITE

Jane is a senior research scientist at the Raspberry Pi Computing Education Research Centre. She was previously a schoolteacher and an IT developer, and now she investigates the teaching and learning of computing (**@janewaite**).

DAD JOKES

Use sequencing and parallelism in Scratch to create characters that can talk to each other

AGE RANGE

7–10 years

REQUIREMENTS

- Scratch 3 (Raspberry Pi or computer version)
- Dad Jokes Scratch studio (**helloworld.cc/dadjokes**)
- Dad Jokes presentation (**helloworld.cc/dadjokesppt**)
- Microphones (optional)

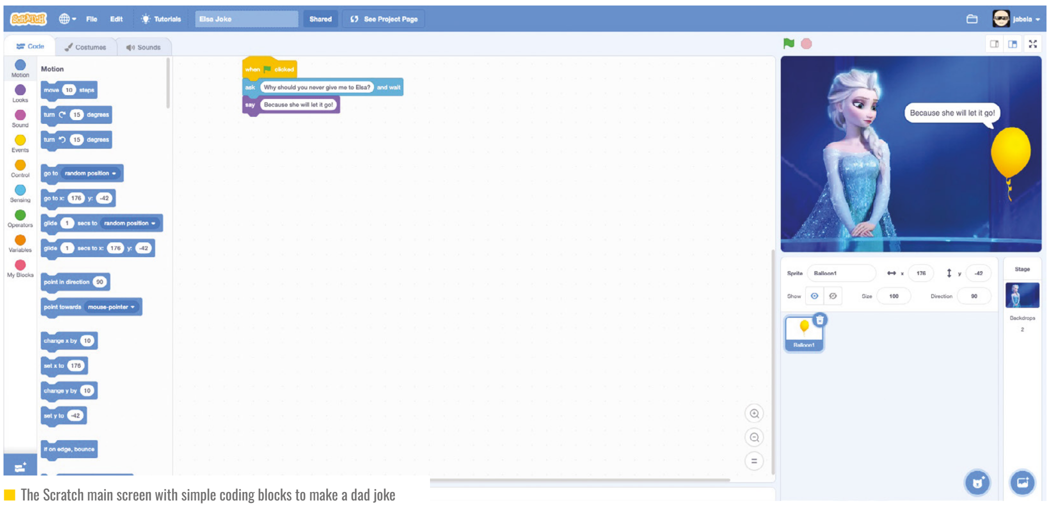
In this lesson, students will learn how to sequence code, understand what different objects are in Scratch, and discover what parallelism is. It's fun because students can tell jokes to each other, and any mistakes made are often funny and enjoyable to fix.

There are also many cross-curricular links, including with English, drama, social-emotional learning, and art. Children can draw their own characters and scenarios and use them for a wide variety of role plays. Not only does this lesson give students a chance to program, but it also gives both children and teachers a way to present a range of social situations in a non-threatening way. (HW)

OBJECTIVES

- ✓ Understand the concepts of sequencing and parallelism in computational thinking
- ✓ Use programming to create conversations and role plays

ACTIVITY 1: PREDICT THE JOKE ANSWER 10 MINUTES



```
when [flag] clicked
ask [Why should you never give me to Elsa?] and wait
say [Because she will let it go!]
```

The Scratch main screen with simple coding blocks to make a dad joke

Show students a simple joke in Scratch that asks them a question and delivers the punchline. They can do this activity on the carpet with individual whiteboards, or in pairs at computers. You could use the example at **helloworld.cc/elsajoke**, or create your own for your class:

1. Display the Elsa joke, or your own joke, to students.
2. Ask students to predict the answer.
3. Show them the answer.
4. If they have used Scratch before, ask students what code is needed to make this type of joke. You could use mini whiteboards to get a whole class response off-screen, or if they are at computers, they could try to find the blocks. Otherwise, show them the code, run it, and ask them what other jokes they think would be good to make.
5. Ask them to either go to the link and create their own jokes by modifying the Elsa code, or create a joke by making their own code from scratch.

>

SCRATCH FOR TEACHERS

If you're using Scratch in the classroom, it's worth creating a teacher account to help you manage younger learners' usernames and passwords (**helloworld.cc/scratchteacher**). There's a helpful article with tips and tricks on page 48 of Hello World issue 18.

RELEVANT LINKS

helloworld.cc/dadjokes: dad jokes Scratch studio

helloworld.cc/dadjokesppt: dad jokes presentation

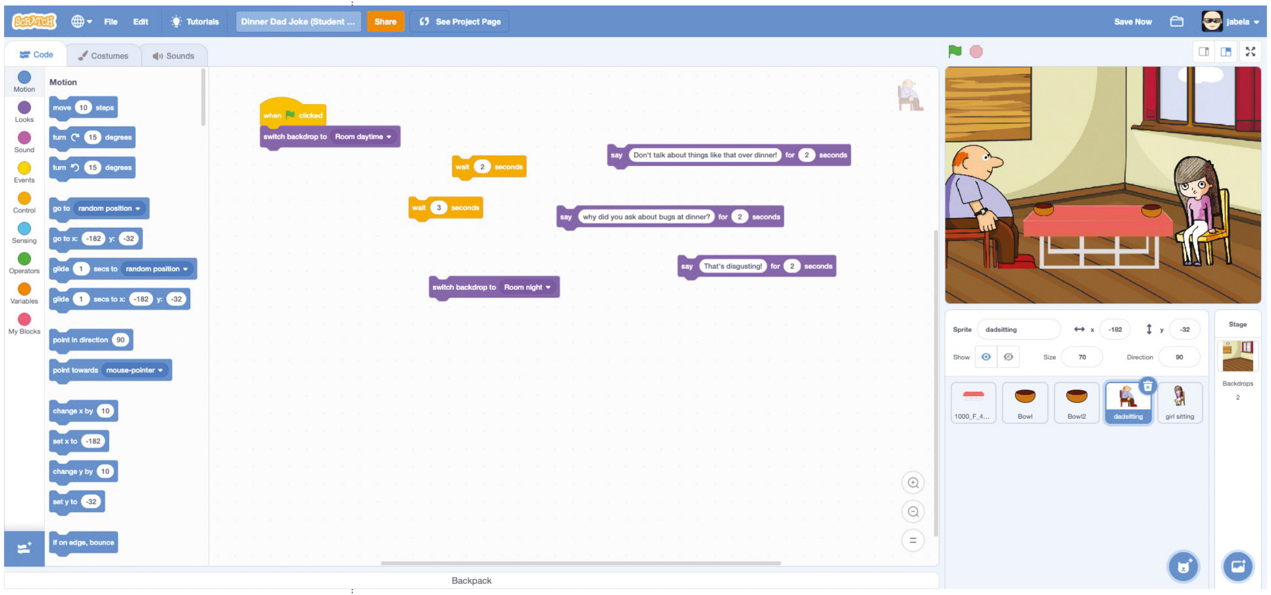
helloworld.cc/sequence: info about sequence for younger learners and why it's important (login required)

ACTIVITY 2: CONVERSATIONAL JOKE 20 MINUTES

You are now going to introduce the ideas of having multiple objects in Scratch; objects doing things at the same time (parallelism); and using timings to create a conversation. For the example at **helloworld.cc/conversation**, I have made a simple dining room arrangement with a father and daughter. This template is flexible, so you can change the characters and furniture.

1. Demonstrate the joke using the teacher version of the file (**helloworld.cc/conversation**).
2. Now show the student version of the joke (**helloworld.cc/dinnerdad**). Ask students to look at the stage and predict what will happen.
3. Run it. You will see that the father will not do anything.
4. Ask students to fix it so that the conversation can happen. This is also a good opportunity to talk about turn-taking in conversations. Each character needs to wait for the other character to finish talking before they talk.
5. Show students the 'dad sitting' sprite and its code. Ask them to investigate and then click the blocks in the code back together in the correct order. Also show them that clicking on the sprites will make them complete their actions, but that the 'wait' command will appear to do nothing (because it is literally just waiting for three seconds).

I chose two seconds for the 'say' commands for easy arithmetic, and because it is quick for students to try out. You can change the time to five or ten seconds for slower readers. You might also want to provide a printed sheet of the blocks that learners can read in their own time. The next task might be less demanding for some students because they can verbally record the joke.



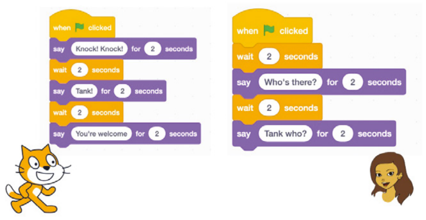
```
when [flag] clicked
switch backdrop to (Room daytime ▾)

wait (2) seconds

wait (3) seconds

switch backdrop to (Room night ▾)

say (Don't talk about things like that over dinner!) for (2) seconds

say (why did you ask about bugs at dinner?) for (2) seconds

say (That's disgusting!) for (2) seconds
```

The blocks are in the wrong order; students need to sort them out and put them together

ACTIVITY 3: RECORDING SOUND 5 MINUTES

Students can now plan their own knock-knock joke and use the 'record' option in the Sounds tab to record themselves delivering their joke. It is quite easy for students to record sound in Scratch, but the classroom needs to be quiet enough so that not too much background noise comes through. Encourage students to plan their script and create a joke. You can show them the project at **helloworld.cc/tankyou** as an example.

If it is too loud or you do not have appropriate microphones, encourage students to create a written version of the joke with a 'say' command and 'wait' statements. It is important that students plan the two characters in parallel.



```
when [flag] clicked
say (Knock! Knock!) for (2) seconds
wait (2) seconds
say (Tank!) for (2) seconds
wait (2) seconds
say (You're welcome) for (2) seconds
```

```
when [flag] clicked
wait (2) seconds
say (Who's there?) for (2) seconds
wait (2) seconds
say (Tank who?) for (2) seconds
```

The blocks of the two characters in the 'Tank You' project

PLENARY ACTIVITY 10 MINUTES

Ask students to share their jokes with each other and to share two things they love about the jokes they see, and one thing that would make the jokes even better if they had the time. If a joke is silent, they might want to perform it for their friends.

You could then finish the lesson with an exit ticket:

- Did the characters on the computer take turns when talking?
- Were they good at it?
- How do we take turns when talking?

DIFFERENTIATION

Support

There are several options to support students:

- Students who find it difficult to type can use the 'record' option to say the words in their joke.
- Students whose first language is not English can use Scratch in a wide range of languages. In the online version of Scratch, there is also an option to translate (click on the 'add extension' button in the bottom-left corner).
- You can use the videos provided in the slide deck to follow along and repeat the building of the code as needed (**helloworld.cc/dadjokesppt**).
- You can print the slides with blocks on to help those who need it.
- For some autistic students, you can use simple conversations and role plays rather than jokes. This is a good opportunity to encourage them to think about conversations they can have.

Stretch and challenge

Students can extend their learning by using broadcasts, a tool that lets students send messages from one sprite to another, to trigger another sprite to do something once a sprite has finished its actions. Students often struggle with this concept at first, because it is not as simple as counting the seconds to synchronise sprites' actions, but it allows objects to take as much time as needed and then let other objects know that they are finished.

The slides include an extension project, with a video to build a knock-knock joke in which an extra object appears (**helloworld.cc/knockbroadcast**). As Scratch projects become more complex, this becomes a useful tool. You could go even further by using the text-to-speech option (see the 'add extension' button in the bottom-left corner) to make bigger projects.

JAMES ABELA

James is the head of computing at Garden International School in Kuala Lumpur, Malaysia. He is a Raspberry Pi Certified Educator, founder of South East Asian Computer Science Teachers Association, and author of The Gamified Classroom (**@eslweb**).

EXPLORING THE KEYBOARD

Introduce learners to the keyboard

The following two lesson plans are taken from the 'Creating media — digital writing' unit of The Computing Curriculum (TCC), written by the Raspberry Pi Foundation. They are aimed at learners aged five to six, and are designed to promote learners' understanding of the various aspects of using a computer to create and change text.

ABOUT THE COMPUTING CURRICULUM

Raspberry Pi Foundation

The Computing Curriculum is the Raspberry Pi Foundation's bank of free lesson plans and other resources that offer educators everything they need to teach learners aged 5–16. It covers the full breadth of computing, including computing systems, programming, creating media, data and information, and societal impacts of digital technology.

Every unit of work contains a unit overview; a learning graph to show the progression of skills and concepts in a unit; and lesson content, including a lesson plan, slides, and formative assessment opportunities. Find them when you sign up for a free account at **helloworld.cc/tcc**.

AGE RANGE

5–6 years

OBJECTIVES

- ✓ Open a word processor
- ✓ Recognise keys on a keyboard
- ✓ Identify and find keys on a keyboard

REQUIREMENTS

- ✓ Colouring pencils
- ✓ Computers with access to a word processor
- ✓ The keyboard template from **helloworld.cc/explorekeys**

This is the first lesson in The Computing Curriculum's 'Creating media — digital writing' unit for learners aged five to six. Learners will familiarise themselves with a word processor and think about how they might use this application in the future. Learners will also identify and find keys, before adding text to their page by pressing keys on a keyboard. (HW)

ACTIVITY 1: CREATING TEXT 5 MINUTES

Ask learners to think about and discuss with a partner all the places they could create and write text. Use pictures to prompt them to think about the different tools that they could use to write, such as a pencil or chalk.

Ask learners to share their ideas with the class. If they do not identify it, explain that they could also create text using a computer.

ACTIVITY 2: WHAT COULD WORD PROCESSORS BE USED FOR? 5 MINUTES

Share an image of a blank word processor document. Ask learners to think–pair–share:

- 'What is this?'
- 'Have you used anything like this before?'

Now, hold a blank piece of paper up to the learners. Referring to the blank word processor document, explain that they are both pages that they can write on. Once again, show learners a blank word processor document and ask them to think–pair–share: 'What could you use this for?'

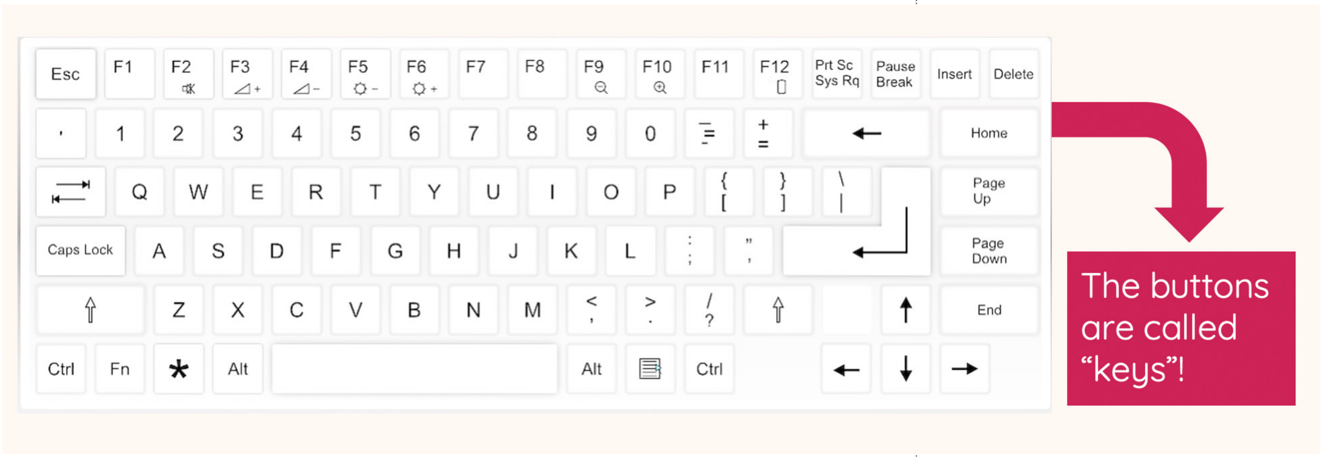


Figure 1

ACTIVITY 3: THE COMPUTER KEYBOARD 10 MINUTES

Ask learners, 'Which part of the computer could you use to add letters to your page?' (answer: keyboard).

Now, share the keyboard template ('A2 Worksheet — the computer keyboard' at **helloworld.cc/explorekeys**) with the learners and explain that the templates need to be kept safe for future lessons. Note: this is a visual representation of a computer keyboard. It has been chosen because the letter keys are labelled with both lower and upper case. The actual keyboards your learners use will look different.

Show **Figure 1**. Ask learners to think–pair–share: 'What do you notice about the keyboard?' Draw the learners' attention to the letter keys, but also allow them to think more generally.

Possible answers include:

- The letters are not in alphabetical order
- The letters are all capitals
- Some of the keys have more than one thing on them

Explain that the buttons on a keyboard are called keys. Explain that their keyboard templates will have capital letters and lower-case letters on to help them find the key that they need. Using their keyboard templates, ask the learners to find and colour in the letters in their names.

ACTIVITY 4: PRESSING KEYS ON THE KEYBOARD

20 MINUTES

Allow learners time to log in to the computers and open the word processor. If this lesson is the first time that the learners will be logging in to the computer, additional support and time may be required for this step.

Once learners have opened the word processor, they can begin to press the keys on the keyboard to write:

- The letters a, b, c, d, e, and f
- Their name
- A friend's name

Ask learners to close their word processor and tell them that they are not saving their work this time. Ask them to log off and shut down their computers.

PLENARY ACTIVITY: WRAPPING UP

5 MINUTES

Ask learners to think about the following questions with a partner:

1. What did you like about using the computer to write?
2. Was anything tricky about using the keyboard?

Ask some of the learners to share what their partner has told them.

RELEVANT LINKS

TCC 'Exploring the keyboard' lesson:
helloworld.cc/explorekeys

ADDING AND REMOVING TEXT

Promote learners' understanding of various aspects of using a computer to create and change text

AGE RANGE

5–6 years

OBJECTIVES

- ✓ Enter text into a computer
- ✓ Use letter, number, and Space keys
- ✓ Use Backspace to remove text

REQUIREMENTS

- ✓ Colouring pencils
- ✓ Computers with access to a word processor
- ✓ 'A1 teacher handout' from **helloworld.cc/addtext**
- ✓ The keyboard template from **helloworld.cc/explorekeys**

RELEVANT LINKS

TCC 'Adding and removing text' lesson: **helloworld.cc/addtext**

This is the second lesson in The Computing Curriculum's 'Creating media — digital writing' unit for learners aged five to six. Learners will familiarise themselves with word processors and how they can interact with the computer using a keyboard. They will focus on adding text and explore more of the keys found on a keyboard. Finally, they will begin to use the Backspace key to remove text. (HW)

ACTIVITY 1: WHAT IS A KEYBOARD? 5 MINUTES

Ask learners what they already know about a keyboard. They may suggest answers like:

- Keyboards have keys on them
- Keyboards are used for typing/writing on a computer
- The letters are not in alphabetical order
- The letters are all capitals
- Some of the keys have more than one thing on them

ACTIVITY 2: TYPING ON A COMPUTER 10 MINUTES

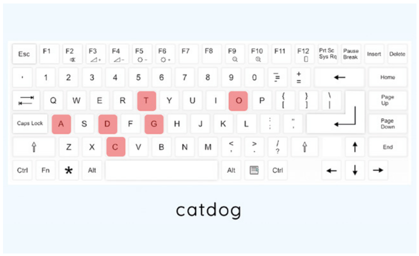


Figure 1

Show the keyboard in **Figure 1** and ask a learner to be the typist. Bring seven learners to the front and give six of them a letter from 'CAT DOG' (you can print them out from the 'A1 teacher handout' at **helloworld.cc/addtext**) and one of them a text cursor card (flashing line).

Ask them to stand in a line with the letters facing away from the class (c a t d o g). Ask the learner with the text cursor card to stand next to the learner holding the 'c' card at the end of the line, with the line on the paper facing the class. To make it appear as if it is flashing, ask the learner to fold and unfold the paper. Explain that the learners need to write 'cat'. Ask the learners to call out the letters that need to be typed. Ask the typist to press the keys on the screen. As a key is pressed, the learner holding the text cursor should step into the space between the next two learners. The learner who has been passed can then turn their letter around to face the class. Repeat this until the learners have written the word 'cat'. Repeat with 'dog'. Keep the learners at the front of the class.

Ask the learners to think–pair–share:

- Is this how you would write 'cat dog' on paper?
- What have you missed out?

Now show the keyboard in **Figure 2**. Ask the learners: 'Does anyone know how to write spaces on a computer?' Share the Space key with the learners and ask them

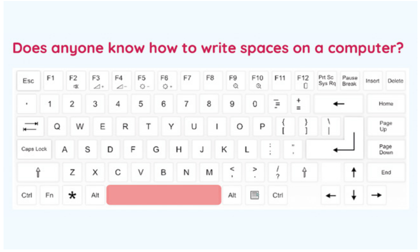


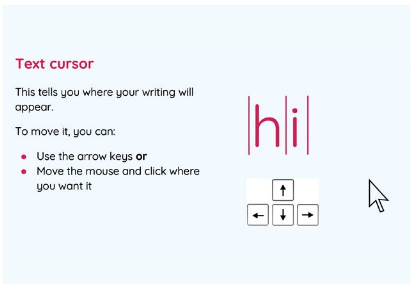
Figure 2

ACTIVITY 2: TYPING ON A COMPUTER (CONT.)

to colour in the Space key on a template of a keyboard (you can download the 'A2 worksheet' at **helloworld.cc/keytemplate**).

Display **Figure 3**. Explain to the learners that the flashing line is called a text cursor. Explain that this tells them where their writing will appear when they write. To move the line, they can either use the arrow keys, or move the mouse to where they want the text cursor to be and then click.

Pretend that you are the mouse, and model moving the learner with the text cursor to between the learners holding the 't' and the 'd'. Ask the learners to press an 'invisible Space key', and ask the learners from 'd' to 'g' to move a step to their left.



Text cursor

This tells you where your writing will appear.

To move it, you can:

- Use the arrow keys **or**
- Move the mouse and click where you want it

Figure 3

ACTIVITY 3: ADDING TEXT ON A COMPUTER 10 MINUTES

Find a picture of a stuffed bear online. Explain that the toy is named Ted and that Ted has been lost. Tell the learners that they can help Ted's owner find him again by writing a description of Ted. This will help people know what he looks like.

Now ask the learners to log in to their computers and open the word processor. You could give learners sentence starters to use when describing the toy. Give the learners time to write a short description of the lost toy on their computers.

ACTIVITY 4: USING THE NUMBER KEYS 10 MINUTES

Ask the learners: 'Does anyone know how to move down to start writing on a new line when using a computer?' Share the Enter key with learners and ask them to colour it in on the keyboard template. Explain that the Enter key is often a funny shape on keyboards.

Next, explain that now that the learners have written a short description of Ted, they need to let people know how to contact the owner. Ask the learners to think–pair–share: 'How will people know how to tell the owner if they find Ted?' They need a phone number. Here, it's important to remind learners not to share a real phone number they might know. This could link to work you have done around online safety.

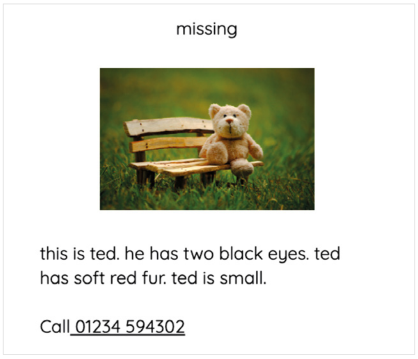
Explain that they need to add a phone number to the bottom of their page describing Ted; suggest to learners that they use 123456. Ask the learners to save their work now, as they will need it next lesson. If your school uses a cloud-based system that allows collaboration on documents (e.g. Google Docs), the learners may not need to save their documents. This is something that you should explain to learners.

PLENARY ACTIVITY: WRAPPING UP 10 MINUTES

Share the Backspace key; explain that it takes away writing, but only removes things on the left (the way the arrow is pointing). The flashing line tells you what will be removed, and it only takes one thing away at a time. Ask learners to colour in the Backspace key on their keyboard templates.

Share an example of the lost-toy poster (**Figure 4**). Now ask learners to think–pair–share: 'What changes would you make to this lost toy poster?' Open the text in a word processor and model moving the cursor using the mouse, and placing it to the right of the error the learners want to change. Explain that this lets them edit their work without having to retype all the words they wrote after the error. Give them time to make changes to their writing using the Backspace key. Reusing the printout from Activity 2, ask four learners to come to the front of the class and hold the letters 'c', 'a', and 't', and place one learner as the text cursor (flashing line) at the front of the word. Explain that you have misspelt the word 'cap'. Ask the learners to discuss with a partner: 'What steps do you need to take to correct the writing?'

Ensure that the learners are using the names of the keys and thinking about moving the text cursor. Use misconceptions to support the learners by allowing and demonstrating mistakes (for example, show that nothing happens if the learners have not moved the text cursor first).



missing

this is ted. he has two black eyes. ted has soft red fur. ted is small.

Call 01234 594302

Figure 4

THE BEBRAS PUZZLE PAGE

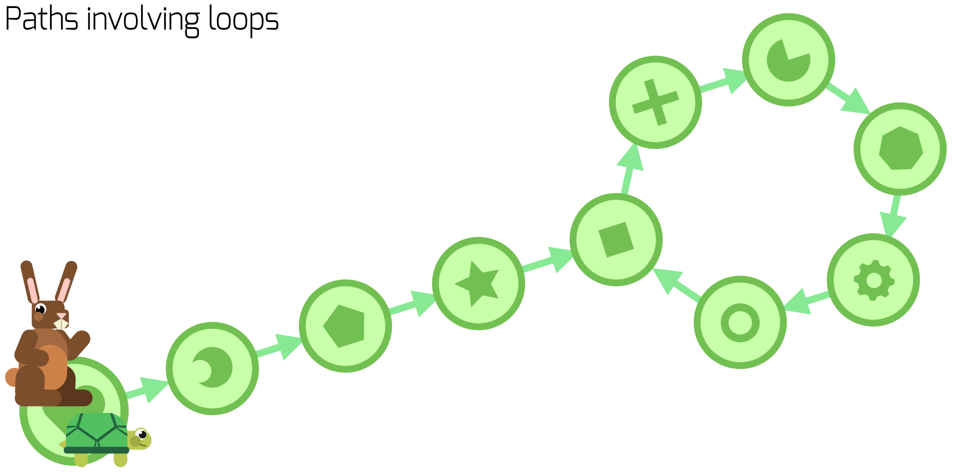
Each issue, **Chris Roffey** shares a computational thinking problem for your students based on the work produced by the International Bebras Community

ABOUT BEBRAS

Bebras is organised in over 55 countries and aims to get students excited about computing and computational thinking. Last November, over 365,000 students participated in the UK annual challenge. Our archived questions let you create your own automarking quizzes at any time during the year. To find out more and to register your school, head to **bebras.uk**.

THE PROBLEM: **TORTOISE AND HARE**

Paths involving loops



DOMAIN

Algorithms and programming

SKILLS

Algorithmic thinking

AGE

12–18 years

DIFFICULTY RATING

Ages 12–14 hard; ages 14–16 medium; ages 16–18 easy

A tortoise and a hare are about to race each other. They both start at the same time, in the circle with a heart in it. They follow the direction of the arrows on the track. The tortoise moves one circle every minute. The hare moves two circles every minute. In which circle will the tortoise and the hare meet for the first time after the start? ***This problem was originally written by the Bebras team from the Philippines. Solution on page 81.***

Further information

Although this is an easy task, it is based on some quite advanced computer science. The task is about traversing a specific data structure with one-way traffic; it's a directed graph with a loop or cycle. An advanced application of this is the analysis of the quality of random number generators, especially those used in the encryption or protection of sensitive data. They usually have a precycle that does not repeat, followed by a cycle. The straight part of the path corresponds to the precycle, while the circular part corresponds to the cycle. Longer cycles make pseudo-random number generators more useful in encryption algorithms, and create stronger and more secure systems that are difficult to crack.

COMPUTING KEYWORD SPOTLIGHT: **CYCLE DETECTION**

Defining words and phrases in computer science

Cycle detection is an important task in computer science. For instance, it can check whether our code is repeating a sequence of tasks endlessly (in an infinite loop), which prevents our program from stopping.

How do computers detect the existence of cycles? One ingenious approach, attributed to Robert W Floyd, is the tortoise and hare algorithm (named after the Aesop fable). As shown in this problem, it features two 'pointers' that move through a path at different speeds, one moving twice as fast as the other. If they meet, we can conclude that there is a cycle. Once we have worked out a way of solving such a problem, it is usually easy to write a program that, in this case, detects the existence of cycles.

WHAT ARE HUMANS FOR IN COMPUTING?

In this new column, **Michael Conterio** and **Tracy Gardner** discuss the evolving nature of humans' interaction with computing systems and look at how we can thoughtfully prepare for the future of computation

It's important to understand the role of humans in computing, as around the world computing is becoming increasingly central to people's day-to-day lives. Many of us will have experienced situations in which a call centre operator says of a reasonable request, "The computer won't let me do that." Why do these situations happen? It's not the computer that has decided what is and isn't allowed, as much as it may seem to be. Everything that a computer system does is, at some level, the result of a decision by a human.

Humans in the driving seat

This means that writing code is not the most important part of computing; deciding what the computer should do is more important. The people who know what a computer should do in a given situation are the subject-matter experts, such as financial experts, scientists, or healthcare professionals. Without those subject-matter experts, the programs written wouldn't be useful and could even be harmful. This is partly why job roles such as business analysts and product managers exist: to bridge the gap between the subject-matter experts and the coders working directly on the system.

Historically, the most important skill for making a computer do what you want has been the ability to code. However, this is starting to change. Increasingly, humans are improving computers' ability to process human language and the information we have already produced on the internet and in data sets, including code. Technologies such as ChatGPT are remarkably good at taking human-language prompts and creating code from them. While they are far from perfect, they allow a person to create code more quickly with input from an AI partner. As technology advances, human tasks will increasingly shift towards deciding what to ask computers to do and evaluating and monitoring their output — not least to ensure that it is ethical.

There is, and will increasingly be, a need for people who can accurately describe the required behaviour of computer systems and evaluate systems to make sure they are performing as expected. There will, of course, continue to be a demand for traditional computer programmers to create specialist software. But more systems will be created by people who understand computation but are not traditional coders.

So what do people, especially young people currently learning, need to know about computers and computation? We'll explore these topics in more detail in future editions of Hello World. (HW)

HUMANS AND COMPUTERS: AN EVOLVING ROLE

Ada Lovelace understood that computers could do more than just manipulate numbers. Instead, they would be able to perform tasks in a wide variety of subject areas, including the arts. However, to do this, humans would have to describe situations and ideas in a way that computers could process. Lovelace is considered one of the first computer programmers, but she wasn't using techniques that look like current programming languages. The way humans instruct computers continues to evolve.

TRACY GARDNER & MICHAEL CONTERIO

Tracy is a computer scientist with extensive experience of computing in industry, research, and education. Michael is a former physicist and now works as an online course production manager at the Raspberry Pi Foundation.

ENGAGING CYBERSECURITY LESSONS

In this Insider's Guide, **Alan O'Donohoe** gives suggestions for fun and engaging activities to bring cybersecurity lessons to life

As computing teachers, it is part of our duty to ensure that children grasp the potential risks associated with cybersecurity. But it's also part of our responsibility to ensure they understand some everyday practical actions they can take to reduce the chances of themselves or others being targeted in an attack and to minimise consequent damage. Cybersecurity education, though, presents some challenges:

- Cybersecurity topics can be dry and dull, making teaching difficult
- The nature of cybersecurity topics means they can sound terrifying
- Teaching cybersecurity tends to be theoretical, as it's not practical to experience an attack
- Students may not perceive themselves as being at risk — they either think they know it all or that they have nothing of value to cybercriminals

There are, however, opportunities for teachers to make learning about cybersecurity engaging and exciting, while also helping students understand the real-world implications of poor online security. In this article, I provide practical examples of activities that you can use to make the teaching of cybersecurity more interesting for your young students. The brute-force attack activity could easily last a full lesson, but the others work better as a smaller part of a lesson, to test and develop understanding.

Teachers can adapt these activities to work with learners of different ages. For example, I've used the first activity with children as young as seven, and have used all these activities with parents and staff members as part of training sessions. You may use them as described first before modifying to suit your preferred teaching style and particular audience. (HW)

ALAN O'DONOHOE
With 30 years of experience teaching and leading in schools in northern England, Alan leads The Exa Foundation (**exa.foundation**) on a mission to inspire digital makers, support the teaching of computing, and promote the appropriate usage of technology (**@exafoundation**, **@teknoteacher**).

BRUTE-FORCE ATTACK

In this activity, through the use of combination padlocks, learners are able to model the principles of a brute-force attack and apply these to password security. I begin by telling the class that I will be distributing ten padlocks around the room and the goal is for them to work together in groups to unlock them. However, before anyone is allowed to touch them, we will discuss strategies they can use.

A brute-force attack is a method cybercriminals use to guess a password by trying every possible combination until the correct one is found. Typically, this is performed by an algorithm capable of guessing thousands of combinations every second. Encourage the students to think about how this concept applies to the padlocks in front of them.

I ask the class to calculate how many possible combinations a four-dial padlock has, and if they were able to test one every second, the maximum amount of time it would take (2 hours, 46 minutes, 40 seconds!). Humans can get tired; they need breaks, make mistakes, and are lazy, so we discuss strategies they could use to reduce this amount of time. Students may ask the teacher questions to reduce the number of possible combinations. I tell them I will never reveal the full code, but I may accidentally give away hints to each team (for example, 'Two of those numbers are correct, but in a different order', or 'No numbers appear more than once'). I remind teams to collaborate by exchanging what they know with other teams to reduce the time required, and stress the importance of keeping and sharing a record of the combinations they have already tried.

It's difficult to predict how long it will take for students to successfully unlock the padlocks. I use the same code for all padlocks. This creates another learning opportunity around the convenience of having only one password versus the risks. It's important to plan for some reflection time at the end to discuss the lessons learnt, the importance of using strong passwords that are difficult to guess to prevent brute-force attacks, and the extra security that multifactor authentication offers.

Children work collaboratively to guess the code for a combination lock using a brute-force method

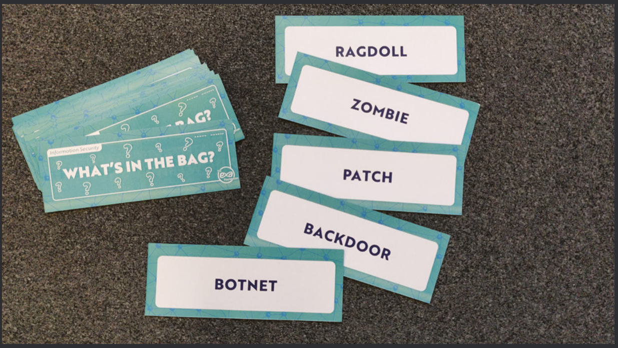
WHAT'S IN THE BAG?

Recommended amount of time: flexible, from 2 minutes up to 20 minutes, depending on the time available. Each phrase selected requires only a minute or two, so it could be used at the end of a lesson with whatever time is available.

This engaging activity is called 'What's in the Bag?' It can develop students' confidence with technical vocabulary while also encouraging them to think critically about cybersecurity risks and strategies in a light-hearted way.

I prepare a bag containing a set of cards, each with a technical term on it connected to cybersecurity. The technical terms represent a mix of threats (potential cyber risks) and protections (security tools and strategies that can be used to reduce cyberattacks). To add a bit of fun, you can also include a few terms that have no obvious connection to cybersecurity. I have used breeds of cats such as Russian Blue, Persian, Ragdoll, and so on. In my experience, students have enjoyed these surprise elements of fun, and there have even been occasions when students have tried to convince me that Russian Blue is a kind of virus or a hacking group!

To begin the activity, the teacher reaches deep into the bag, adding an element of drama and surprise, as they select a phrase and read it aloud. Student players have to respond quickly to correctly identify the phrase and categorise it as threat, protection, or cat (or other distractor). It also works well when students are organised into teams that compete against each other, with a point awarded when teams correctly identify whether the term is a threat or a protection. The teacher may select individual team members in turn to respond initially, then allow the team to discuss the details before scoring additional points for an explanation. If teams correctly identify that the term is a distractor, they score a bonus of three points. After correct identification, the teacher may ask teams, for additional points, to explain what the phrase means and how it relates to cybersecurity. For example, if the phrase 'phishing' is selected, the students would need to identify it as a threat and explain what a phishing attack is, the methods used by perpetrators to conduct such an attack, how a potential victim can spot a phishing scam, and steps they could take to protect themselves against it.



Prepare a bag containing a set of technical terms connected to cybersecurity. Can you spot the breed of cat?

Occasionally, I filter out some terms before starting the activity so that the remaining collection reflects the ability of the teaching group. The set I prepare includes a combination of complex technical terms and more obvious terms, so that there's something for everyone.

>

PACKET EXCHANGE

Recommended amount of time: 20–30 minutes. This activity is worth repeating so that students get the hang of it.

The teacher prepares a set of double-sided cards, with one side featuring a phrase related to cybersecurity and the other featuring a comprehensive definition of that phrase.

Students are then paired up and take turns asking their partner to define the phrase on their card. If the partner being quizzed needs help, the questioner can use the definition on the reverse of the card to provide support in answering the question. However, the questioner needs to judge how subtle or strong the help they provide should be, depending on how successfully the quizzed partner is able to define their phrase. There is skill in providing just enough support so that the person answering is able to arrive at the definition themselves without too many prompts. After each student has quizzed their partner, they swap cards with each other, or find a new partner, and repeat the activity. This allows students to learn from each other and reinforce their understanding of key cybersecurity concepts.

From experience, I've found that students have enjoyed learning about cybersecurity in this way because it encourages collaboration and support, and develops understanding in a non-threatening way. By working with their peers, students are able to reinforce their understanding of key concepts and develop the skills they need to stay safe and secure online.

By working with their peers, students cam reinforce their understanding of key concepts

There are plenty of opportunities for teachers to make learning about cybersecurity engaging and exciting

CYBERSECURITY SONGS

Through my work with The Exa Foundation, I've been visiting primary and secondary schools across England to help school communities (not just children) develop their understanding of cybersecurity risks. Visiting different learning settings helps me develop different approaches all the time as I seek to find new and engaging ways of presenting the content. One of my latest endeavours has been to rewrite the lyrics of popular songs, adapting the messages to a cybersecurity theme. I'm fully aware that I have a terrible singing voice, and the lyrics still need some work. However, I have noticed that the children respond very positively!

I ask the class to list all the cyberthreats or protection strategies they can identify while I perform the song, then we review them to see how many they spotted correctly. Finally, I'll share the lyrics on screen, and some classes like to join in the song for a second rendition. *Attack, Survived* is one of my latest examples and I perform it to an instrumental/karaoke version of Gloria Gaynor's *I Will Survive*. If you'd like to know more about the other songs, get in touch!

ATTACK, SURVIVED

At first, I never knew, till the day you tried
To brute force my account, then my network died
Soon came the phishing scams, they all strung me along
They did me wrong, I should have made my passwords strong

Then a DDoS attack, emoji evil face
Found your digital footprints after running a network trace
I should have changed those stupid passwords, used multifactor too
If I'd known about your cyber tricks, I'd have never trusted you

CHORUS
Log off now, go, forever more
You're locked out now, I've shut down that back door
Weren't you the one who tried to cause a data breach?
Now you've been rumbled, it's time you said goodbye

Next time you try, attack denied
Automated backup strategies will help protect my files
I've got all my firewalls on, all my data's encrypted strong,
Attack survived, attack survived

Port scans helped identify potential risks
Closed them down, reduced exposure to your hacker tricks
I spent oh so many nights building my network defence
I used to be clueless, but now I've developed cyber sense

And you see me, somebody new
I'm not that careless little n00b who once trusted you
You thought zero day made me a target, then you failed to see
I quickly patched my software to reduce vulnerabilities

CHORUS
Go on then try, attack denied
Regular antivirus updates, protect all my drives
I've got auto updates on, I've made all my passwords strong,
Attack survived, attack survived

TEACHING PRIMARY PROGRAMMING WITH SCRATCH

Scratch has been the go-to language for novice programmers since 2007. Today's landscape is very different, and is elegantly brought together in Phil Bagge's latest book

INFO **BY** Phil Bagge | **PUBLISHER** University of Buckingham Press | **PRICE** $29.75 | **ISBN** 9781915054203 | **URL** **helloworld.cc/scratchbook** Matthew Wimpenny-Smith

I have been teaching since before the development of Scratch and I'm now involved in initial primary-computing teacher training. I would say that this new programming book from Phil Bagge is an absolute must-read for all primary teachers. The book provides trainee teachers with an accessible and easy-to-understand guide to teaching primary programming using Scratch. For expert computing teachers, it gives a solid refresher on the current computing pedagogy research and how it can be applied in the classroom. In the introduction, this is highlighted by different suggested entry points for using the book, depending on your current level of knowledge and experience.

Phil is a UK-based computing educator, author, school inspector, and advisor, and he was involved in developing the national curriculum for computing in England. This latest book is part of a growing library of his previously published Scratch books (for example, **helloworld.cc/scratchbook2015**). Much like the earlier series, the main *Teaching Primary Programming With Scratch* teacher book has four accompanying pupil books and two home-learning books, which are jam-packed full of lesson activities, resources, and guidance. Together, these formulate a complete and comprehensive scheme of work for Scratch programming for learners aged seven to eleven, based on Phil's new Code-it Gold scheme of programming (freely available at **code-it.co.uk/gold**).

Backed by research

One of the main differences between this book and Phil's previous work, apart from the Scratch version change, is that it is informed by the latest pedagogical research in computing education. For example, it weaves in activities using code comprehension (getting children to read code before they write it), Parson's Problems, design in programming, and the PRIMM (Predict–Run–Investigate–Modify–Make) approach to programming. The book also links these pedagogies to the educational theories of cognitive load and constructionism. The accompanying pupil books continue in the same style and form a basis for putting the methodologies and research from Phil's book into practice. The pupil books are structured in the form of photocopiable pupil resources on one page and teacher hints and tips on the other. The teacher pages provide additional narrative and annotations to support their pupils' learning.

Like all of Phil's books, this is written in an engaging and accessible way so that even novice computing teachers can access it. For example, throughout the book there are plenty of everyday classroom examples, Scratch code snippets, illustrated models, and references to the research that he has used to inform the work. The book has a logical structure, opening with a section on concepts and followed by chapters on pedagogy, processes, and where to go for more support. As Sue Sentance writes in the foreword: "What a treasure trove!" (HW)

UNLEASHED

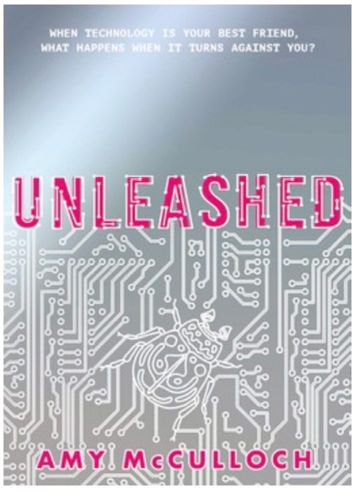
Strong girl characters, teen coders, sentient robot pets, and a corporation with questionable ethics makes for an exciting teen drama

INFO **BY** Amy McCulloch | **PUBLISHER** Simon & Schuster **PRICE** $9.96 | **ISBN** 9781471169984 | **URL** **helloworld.cc/unleashed** Katie Sparks

The second in the *Jinxed* series — something I didn't realise when I picked it up at the library — *Unleashed* is a really fun book for young adults. It explores big ideas such as digital literacy, ethics in the tech world, and access to technology and opportunities. The book is an exciting teen drama, which is a fun way of introducing students to the topics of the impacts of technology and how individuals still have agency.

In the *Jinxed* world, most people own a personal robot pet, or a baku, with customisation options from basic to expensive. Similar to mobile phones, the bakus manage your communications, calendar, money, and so on, and are subject to software updates. Following a forced update from the tech pets' parent company, MONCHA, people start behaving strangely. What's more, the roll-out is initially targeted at basic models, ringing alarm bells about MONCHA's ethics. Enter our heroes: a group of kids, with their almost sentient bakus, trying to uncover what's happening at the heart of MONCHA and put things right before it's too late. As the plot unfolds, the teenagers question their views of bakus and MONCHA, and ask who they can really trust. The book is a fun way to unpick how we can too easily rely on technology and should further question anything advertised to us.

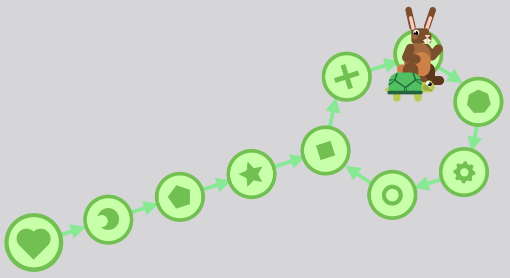
I really liked the girl characters in this book, such as Lacey, who is an aspiring engineer. The characters are smart and driven, yet they're also everyday role models. There are lots of moments of teenagers being teenagers: discussing who's interested in who, getting ready for a party, or making weekend plans. When the boys suggest that the girls can't do something "because it's not safe", the girls advocate for each other. These relatable characters are more than just the sum of their knowledge and abilities. Best of all, the characters all have their own thing: their own tinkering, coding, and problem-solving style. But can the kids get past their differences and save their family and friends? Pick up *Unleashed* to find out — I'll definitely be looking out for others in the series. (HW)


WHEN TECHNOLOGY IS YOUR BEST FRIEND, WHAT HAPPENS WHEN IT TURNS AGAINST YOU?
UNLEASHED
AMY McCULLOCH

BEBRAS PUZZLE SOLUTION (PAGE 74)

The tortoise and hare will meet again after six minutes on the circle shown below.

After one minute, the tortoise will be on the moon and the hare on the pentagon. After two minutes, the tortoise will be on the pentagon and the hare on the square. The hare will now turn left and start going around the loop continuously. After three minutes, the tortoise will be on the star and the hare on the heart. After four minutes, the tortoise will be on the square and the hare on the cog. The tortoise will now turn left and start going around the loop continuously. After five minutes, the tortoise will be on the cross and the hare on the square. After six minutes, the tortoise and the hare will both be on the heart, and so will have met for the first time since the start of the race.

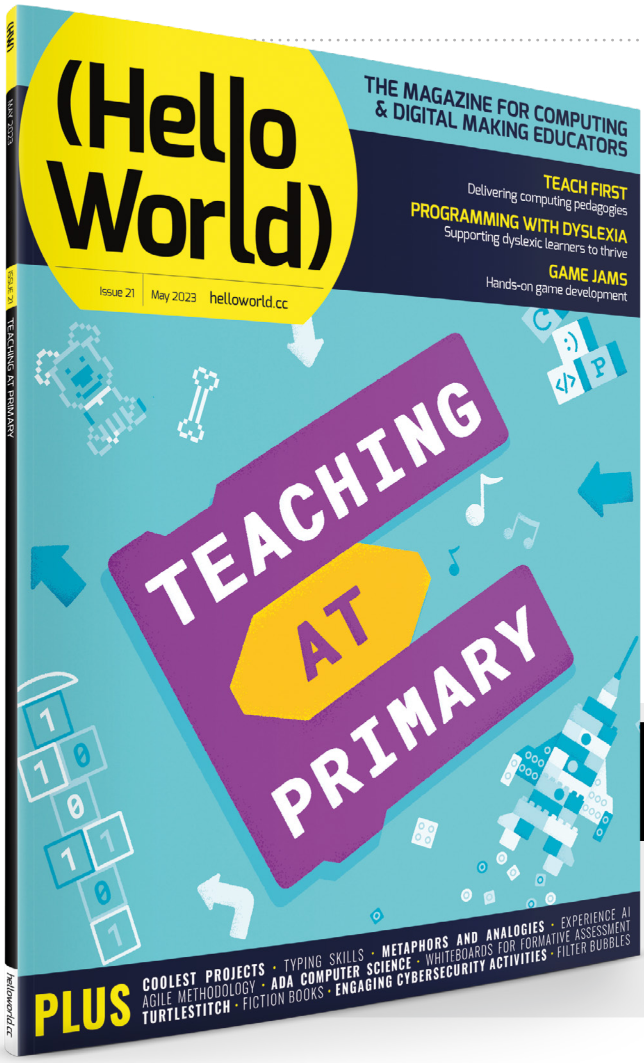


"HELLO, WORLD!"

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.


(Hello World)
THE MAGAZINE FOR COMPUTING & DIGITAL MAKING EDUCATORS
Issue 21 | May 2023 | helloworld.cc
TEACH FIRST
Delivering computing pedagogies
PROGRAMMING WITH DYSLEXIA
Supporting dyslexic learners to thrive
GAME JAMS
Hands-on game development
TEACHING AT PRIMARY
PLUS COOLEST PROJECTS · TYPING SKILLS · METAPHORS AND ANALOGIES · EXPERIENCE AI
AGILE METHODOLOGY · ADA COMPUTER SCIENCE · WHITEBOARDS FOR FORMATIVE ASSESSMENT
TURTLESTITCH · FICTION BOOKS · ENGAGING CYBERSECURITY ACTIVITIES · FILTER BUBBLES

Q WHO MAKES HELLO WORLD?

A Hello World is the official magazine of the Raspberry Pi Foundation.

Q WHY DO 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 magazine is available three to four times per year. Check out our podcast (**helloworld.cc/podcast**) and monthly newsletter (**helloworld.cc/newsletter**) to get more great 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
SEPTEMBER 2023
THEME: **AI**

(Hello World)

helloworld.cc

Raspberry Pi
Foundation