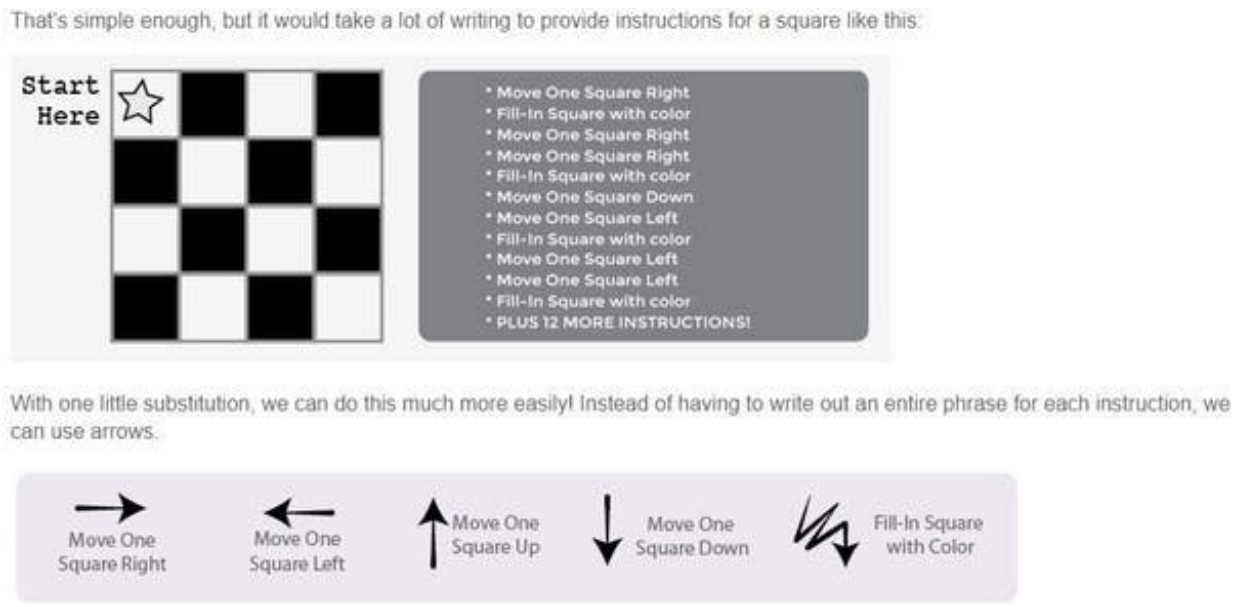
Why a computer isn't the first thing you need when teaching kids to code

December 8 2020, by Lynda Colgan



Graph paper programming from Learn to Code. Credit: Learn to Code, CC BY

In 2020, COVID-19 hit, and globally, there was a massive shift to online instruction. Educators and parents realized that going forward, now that school boards have invested heavily in remote learning it has become a core aspect of education.

Some people have surely been impressed by what even the youngest learners can do with technology. At the same time, it's also become clear

that our societies are going to need a generation of experts who can create whatever will supersede the huge tech platforms of our day like Google or TikTok.

But success in teaching children coding is not about choosing the most cutting-edge technologies or programs. It is about all the learning that is critical as a foundation. Children must learn to locate and orient themselves and other objects in space, and how to visualize such movements and relationships. They must also learn how to communicate and how to solve problems.

Human cognition drives code and, by extension, the computer. It's the human cognition manifest in wondering, learning to pose problems and "debug" solutions that we should be most concerned with.

Roots of coding

A practical pre-coding resource, aptly named *The Roots of Coding*, is under development (for early 2021 release) by the experienced educators from the Mathematics Knowledge Network's Early Math focus team, which I lead. This network is a coalition of Ontario-based academic researchers, school district and community partners, including some specialized in Indigenous math education.

With this resource, educators will find a guide for drawing on music and dance, acting games and other multi-sensory approaches to help children learn how to talk to their peers. The source provides curriculum for guiding children in learning to move their own bodies in relationship to others—and then, how to build models based on instruction.

This resource teaches foundational coding principles while children are mastering other linguistic and spatial skills.

Computational thinking

At its core, computational thinking is about communication. Computers act in response to instructions. If you think conveying clear directions is simple, watch the "Exact Instructions Challenge" with YouTuber Josh Darnit and his children. They attempt to to make a peanut butter sandwich by following apparently direct instructions with precision.

To learn computational thinking, children must learn to make their abstract problems, knowledge, processes and solutions clear and systematic enough so that they are available to other students, open for discussion and debate. They must learn how to express and share reflective thinking processes through spoken and written words and diagrams.

Existing coding support

In Canada, we have been fortunate to have a national program that funds coding instruction.

The program has funded projects such as CanCodeToLearn, Hackergal and Black Boys Code. These provide kindergarten to Grade 12 students and teachers with opportunities to learn digital skills including coding, data analytics and digital content development. These complement and supplement provincial curriculum.

Programs like these employ programmable robots and free, user-friendly computer languages like Scratch and Lynx to bring coding into homes and classrooms.

Such programs make it possible for even the youngest children to successfully instruct a computer to act in a specific way (the definition of computer programming) by following a particular sequence of

instructions.

Lessons from LOGO

More than four decades ago, computer scientists at MIT developed the programming language LOGO to teach children the fundamentals of computer programming. Children could explore how commands work through drawing geometric shapes, solving math problems and creating games. LOGO was introduced to schools worldwide.

But as teaching and learning researcher P. Gibbons notes in the essay "Logo Learning: What the Learners Say," in the collection *Learning in Logo Microworlds*, there was some notable resistance from educators and parents who thought learning LOGO would have no lasting benefit because so-called "real" programmers didn't use LOGO and it was just for kids.

Today, Andreas Schleicher, director of education and skills for the OECD, has suggested that teaching children coding is a waste of time because coding is a "technique of our times" that will quickly become obsolete.

What such criticisms of teaching LOGO and coding miss is that the benefit of learning either at school isn't foremost about early workforce technical instruction. Rather, it's about children learning thinking and problem-solving that will be transferable.

Learning to code is like learning to read: recognizing letters and symbols, sounding out words and making sentences. But the true power of learning to read is actualized when students can read to learn. Similarly, the true power of learning to code is when children apply the thinking and problem-solving they learn through coding, what might be called coding to learn.

Tactile learning, embodied games

There are ways to prepare children for coding that are accessible to classroom teachers with limited (or no) knowledge about coding.

Games such as four squares, hopscotch and Simon Says are about learning spatial orientation or visualization in relationship with others and how to follow instructions.

For young children, computational thinking activities begin in concrete and tactile ways. For example, children can be instructed to make a shape, pattern or letter on grid paper:

When children are challenged to break down a complex task of their own choice, they intentionally take charge of their own learning. Cultivating this impulse is a central tenet of bolstering student agency and a fundamental skill in constructing knowledge.

By focusing on the process of generating, concretizing and evaluating their ideas, children can become more skilled at thinking critically and creatively, and building more elaborate and applied mental schema.

When students can recognize that there are multiple correct solutions, and many unique and creative answers, this results in productive collaboration. It leads to new opportunities to play with and "chew on" emergent, collective and creative ideas.

In the end, it's today's creative problem solvers who will be the true powerful technologies of the future.

Provided by The Conversation