Coding in the Ontario Mathematics Curriculum, 1-8
Might it be transformational?

Gadanidis, G., Floyd, S., Hughes, J.M., Namukasa, I.K. & Scucuglia, R.

Notes: (1) We use the term *coding* to match the language used in the Ontario curriculum, and to generally refer to the computer science concepts and processes that have been added to the Ontario mathematics curriculum, 1-8. (2) The ideas below are a synopsis of a journal paper we are in the process of authoring.

IN BRIEF

Is it possible that coding may, in some contexts and to some extent, fundamentally transform how we learn, teach and think about mathematics, not just for a few people here and there, but on a grander scale?

- Coding is a computational literacy in our society, used by many fields in academic and professional settings.
- The recent Ontario grades 1-8 mathematics curriculum gives access to this literacy to all grades 1-8 students.
- This creates a significant opportunity for new ways to remediate, reformulate, reorganize and revitalize mathematics education for students and for teachers.

Algebra in mathematics

diSessa (2000, 2018) studied the mathematics Galileo used to investigate uniform motion. One of the "difficult" problems that Galileo was trying to solve related to distance = rate x time (d = rt). For example, if you are travelling at the rate of 60 km/h, for 2 h, then the distance you travel would be 60 km/h x 2 h, or 120 km. This problem was difficult for Galileo because algebra was not yet developed, and he relied on convoluted proportional reasoning to understand it and explain it.

Once algebra became accepted and widely used in mathematics, Galileo's problem was no longer difficult. The idea that distance = rate x time (d = rt) is easily accessible to students today, because algebra transformed how and what mathematics may be done, who may do it, and how it may be taught and learned.

Coding in the new Ontario math curriculum, 1-8

In 2020, a significant change occurred in the Ontario curriculum, with the integration of specific coding expectations in the algebra strand, for each of grades 1-8, with the goal that students "solve problems and create computational representations of mathematical situations by [reading, altering,] writing and executing code" (OME, 2020). Algebra appears to be a logical choice for coding integration, as coding often relies on algebraic representation.

Below is a concise list of the new coding goals and coding skills introduced as expectations in the Ontario mathematics curriculum, 1-8 (OME, 2020):

Overall	Solve problems and create computational representations of mathematical situations using coding concepts and skills		
Verbs	Read, alter, write, execute		
GRADE	**CODING CONCEPTS**	**GRADE**	**CODING CONCEPTS**
1	Sequential events	**5**	Conditional statements
2	Concurrent events	**6**	Efficiency of code
3	Repeating events	**7**	Defined count and/or subprogram
4	Nested events	**8**	Analysis of data

Table 1. Coding expectations in the Ontario mathematics curriculum, 1-8

Is it possible that the introduction of coding in Ontario mathematics education may play a role that is similar to the role the introduction of algebra played in mathematics? Might coding, in some ways and to some extent, fundamentally transform how we learn, teach and think about mathematics in grades 1-8 education? To answer this question we turn to diSessa (2000, 2018), to view coding through a *literacy* lens, and map the Ontario situation to diSessa's literacy *principles*.

Coding as a literacy

diSessa (2018) identifies 5 principles that synthesize the meaning and effect of a literacy. We briefly define and discuss each of these principles below, in relation to the Ontario context:

1. **"A literacy is a massive social/intellectual accomplishment of a culture or civilization, where many competing forces, over decades or centuries, eventually settle on a particular representational form for wide-spread learning, use, and subsequent value." (diSessa, 2018, 7)**

Coding is a literacy in our society today. Many fields have a coding-related side: computational biology, computational mathematics, computational finance, and computational medicine, to name a few examples. Barba (2014) notes that the authentic computational practices of scientists and professionals involve solving real-world problems and building knowledge through computational "conversation" and "interaction" with their field.

This does not mean that all of society is affected in this way. As diSessa notes about algebra:

> It is not much good for many cultural functions, for example, for poetry or courting. But it is useful enough, for enough important things, that our prospering and even surviving as a civilization (what is the change in rate of change of global warming?) are implicated. (diSessa, 2018, 7)

In the case of education in Ontario, the formal and very explicit and specific addition of coding to the mathematics curriculum (see Table 1) suggests that coding may be seen, to some extent, as a potential literacy in this context, affecting almost 4,000 schools, approximately 1.4 million students, and over 80,000 teachers.

To get a better understanding of the potential effect of coding as a literacy, we examine the Ontario context through the lens of each of the other 4 literacy principles identified by diSessa, by looking at Ontario curriculum documents, teaching resources developed by team members, and data from mathematics + coding classroom research conducted by team members in

Ontario and in Brazil.

2. *Remediation* of concepts, problems and processes, thorough the new dynamic representational system.

The remediation of mathematics concepts, problems and processes is an explicit curriculum direction in Ontario, as students in each of grades 1-8 are expected to "solve problems and create computational representations of mathematical situations by writing and executing code" (OME, 2020), altering the "representational infrastructure" (diSessa, 2018, p. 25) available to teachers and students. This is further supported by the listing of specific coding concepts and processes to be used in each of the grades (see Table 1), which offer dynamic representations and models of mathematics concepts and relationships. Writing more broadly about possible effects in our society, diSessa (2018) notes:

> Dynamic and interactive representations on computers, along with the ability to design and enact specialized representations on demand and often quickly means that intellectual changes easily on the scale of what algebra or calculus brought us are almost certainly in the offing. (p. 25)

3. *Reformulation*, through cognitive shift, seeing mathematics though a new perspective, with a potential to reveal cognitive simplicities.

diSessa (2018) notes that reformulation is

> a domain-by-domain issue. One needs to find the productive roots of different ways of thinking about each domain. Certainly there are some generalities, but so far as I have seen (and a lot of literature agrees), one needs to understand how learners construe particular domains, and how they may profitably construe them differently. (p. 27)

What might reformulation look like in the Ontario context? Let's start with a simple example involving inequalities in grades 4-8, and consider how coding may reformulate teaching and learning.

In grades 4-8, Ontario students study inequalities: they solve inequalities, including ones that involve multiple terms and operations, whole numbers as well as integers, and verify and graph solutions. The traditional pedagogical approach would involve:

- inequalities such $x > 5$, $2x > 8$, $2x + 3 > 7$, $3x - 5 > x + 1$ and $2x - 10 > -24$
- algebraic manipulation of inequalities to simplify them and solve them, and
- representing solutions as lists of numbers, as sets, and as number-line graphs.

However, starting in grade 5, students are also expected to use conditional statements in code to represent mathematical situations, which is a natural fit with the topic of inequalities. The examples below (Gadanidis 2021a) illustrate some simple ways that traditional thinking about inequalities may be reformulated when we have access to a coding environment.

Example 1: Here is a simple coding + inequalities task, using only 3 lines of Python code:

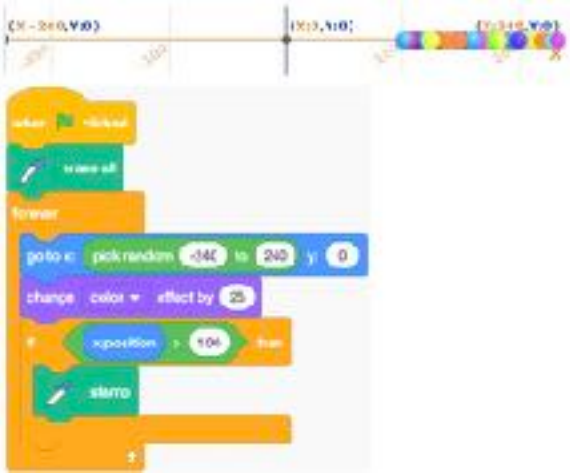
- The Python code on the right solves the inequality $x > 5$ (for numbers 1-10) and lists the solution.
- How would the solution change if we edit line 2 of the code in each of the ways shown below?

```
1 for x in range(1,11):
2     if x > 5:
3         print(x)
```

```
6
7
8
9
10
```

```
2    if 2*x > 10;
2    if 2*x + 5 > 15;
2    if 3*x + 5 > x + 15;
```

- Noticing that the solution does not change, students may then edit line 2 of the code in new ways to create other equivalent inequalities.



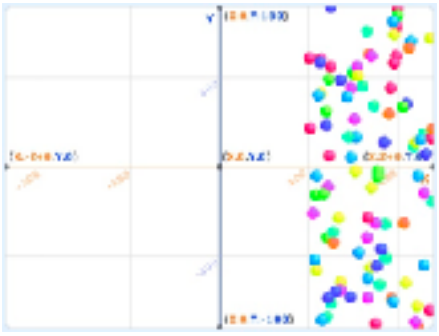
This is a very different learning experience than say a Socratic lesson where the teacher shows, step-by-step, how to edit inequalities, to make them more complex-looking or to simplify them, and to solve them and then students practice what the teacher demonstrates for similar examples. Using the 3 lines of Python code above, students may quickly edit the code, run the code to get immediate feedback, and learn both coding and mathematics concepts, playfully and incidentally.

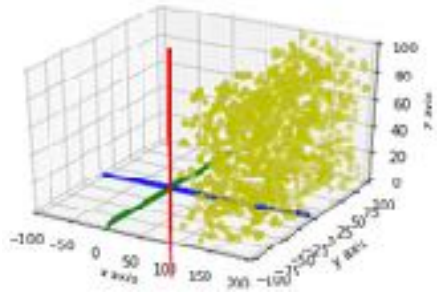
Example 2: Shown above right is a way students may use Scratch to plot number-line graphs of inequalities.

- This Scratch code plots a number-line graph solution for the inequality x > 100.

Example 3. The Scratch code in Example 2 may be edited to plot the solution in 2 dimensions. Notice that:

- The solution to x > 100 depends on the assumptions we make.
- If we assume a 1D context, the solution lives on a number line, as in Example 2.
- If we assume a 2D context, the solution lives on a plane, as shown on the right.
- If we assume a 3D context, then the solution lives in space, as shown below right (plotted in Python).





In such contexts, teachers notice that math + coding tasks, where students have opportunities to read, alter, create and execute code, and see what changes, leads to different types of learning:

— My biggest learning was about incidental learning. Coding facilitates that.

— It's really neat because it extends their thinking, but in a natural way.

Students commented:

— It was funny. I accidentally did something cool. Using trial and error.

— I did something I didn't know how to do.

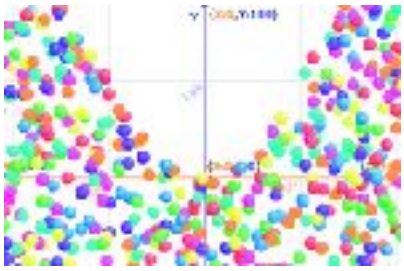
— It's exciting. What might happen next?

— One little thing, one little change, can turn into a big idea.

The reformulations of mathematics concepts and relationships in the above examples also point to some cognitive simplicities. For example:

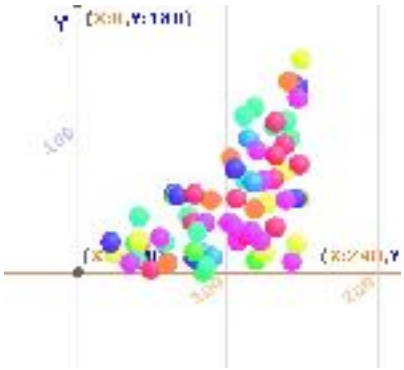
- It would be cumbersome to do the above tasks with paper and pencil
- The ability to dynamically model concepts and relationships, and in more than one way at

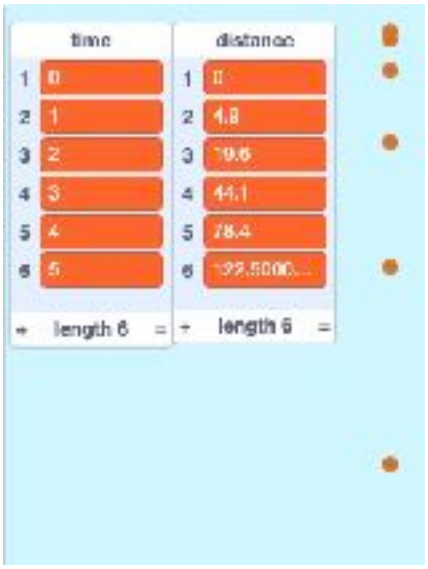
once (such as a list on numbers and their plot on a number line), facilitates the examination of related instances of a relationship with ease
- The ability to make expressions like 2x + 3 > 15 the object of code, gives abstract concepts, relationships and representations a tangible feel, especially when you can also represent them in new ways (lists, plots).
- It is easier, if not natural, to extend concepts, to consider "what if" questions, and to experience new and unexpected mathematics ideas and relationships (for example, inequalities plotted in 2D and 3D).



This may be extended further. For example (Gadanidis 2021b):

A. Conditional statements may involve Boolean operators, such as *and*, *or* and *not*. What does the "solution" look like as a list of numbers or as a 1D, 2D or 3D graph?
B. Puzzles may be created where students write code and conditional statements that match a 2D plot of a defined region, such as a square or triangle. (see plot above right)
C. Students may wonder about boundaries that are not linear. How do we make a line curve? (see plot on the right)
D. Students may wonder about plotting increasing/ decreasing patterns from real-live, such as the distance travelled by a falling object over time, where the rate of change in not constant? (see image on the right)





4. *Reorganization* of the intellectual landscape, changing who gets to do what and when.

It is interesting that the algebra strand expectations in the 2020 curriculum changed in ways that appear to complement the new coding focus. It is as if the curriculum writing team was thinking about coding expectations and at the same time considering (a) mathematics concepts needed to use them and (b) mathematics concepts where there might a best representational fit. For example:

- Repeat and nested coding structures were placed in the same grades as repeating patterns;
- understanding symbols as variables now starts in grade 2 rather than grade 4 matching the natural use of variables in code; and
- solving inequalities, which was not in the previous curriculum, was added starting in grade 3, which sets the stage for using conditional statements in code starting in grade 4.

The apparent need for meaningful connections, between coding concepts and mathematics concepts, appears to have led to a reorganization of the previous curriculum, giving younger students greater access to what were previously more advanced mathematics concepts in higher grades.

In addition, at the classroom level, access to coding changes what and how students may do mathematically, and what control they have over the learning sequence, as illustrated by the student and teacher comments below:

— I'm going to make my own [code]. I'm not going to copy what's on the screen. I'm going to do something new. Then I'll call you and say "Watch this!" [grade 3 student coding for the first time]
— It feels like there's more space. You don't have to do it like everyone else. It lets you go in depth. You see how each function affects the next. How it connects together. [grade 10 student in an integrated mathematics and computer science class]
— It also allows students to be more independent. Working on a program, errors are immediately fed back to you, and you are able to recognize and correct them, as opposed to waiting for a teacher. [grade 10 teacher]

5. *Revitalization* of the field, and how it is taught and learned.

It is typical in the classrooms we work in that teachers express that they are pleasantly surprised that math + coding appears to be especially beneficial for students that underachieve, that students generally take to coding easily, and that students incidentally access mathematics ideas that are above their grade level.
— I wish you were here to see the kids that never do well in assessments. I've never seen that part of him. Words coming out were impressive.
— I found that sometimes the tasks we might feel initially [to be] difficult, the kids got just like that. It has made me less fearful to go beyond the curriculum. In Grade 4 you're not supposed to learn this. Well, what's stopping us from showing them a little bit beyond that?

The learning atmosphere also appears to be revitalized:
— I didn't take this course expecting it to be more collaborative. It just happened. Naturally. I like it.
— It's more of a group feeling in the atmosphere, asking questions and trying to understand. It feels good to know if you're stuck you can turn to someone else for help.
— It has helped me with my collaboration. I'm more open to work with people.

Concluding comments

Coding is a computational literacy in our society, used by many fields in academic and professional settings. The recent Ontario grades 1-8 mathematics curriculum gives access to this literacy to all grades 1-8 students and creates a significant opportunity for new ways to remediate, reformulate, reorganize and revitalize mathematics education for students and for teachers.

References

Barba, L.A. (2014). Computational thinking is computational learning. Keynote address at SciPy (Scientific Computing with Python) Conference, Austin, Texas. Video retrieved 5/01/17: http://lorenabarba.com/gallery/prof-barba-gavekeynote-at-scipy-2014

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy.* Cambridge, MA: MIT Press.

diSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. Mathematical Thinking and Learning 20(1), 3-31.

Gadanidis, G. (2021a). *Understanding math + coding, 1-8.* Worldiscoveries, Western University: London, ON.

Gadanidis, G. (2021b forthcoming). M*ath + coding puzzles, 1-8.* Worldiscoveries, Western University: London, ON.

Ontario Ministry of Education (2020). *The Ontario curriculum, grades 1-8: Mathematics.* Queen's Printer for Ontario: Toronto, ON.