CODING for YOUNG MATHEMATICIANS

George Gadanidis, PhD

ABOUT THE AUTHOR

George Gadanidis spends on average 50-60 days annually in K-8 classrooms, collaborating with teachers to design better ways of engaging children with mathematics. George is Professor of Education at Western University and a Fellow of the Fields Institute for Research in Mathematical Sciences.

Here is more of George Gadanidis' work on Math + Coding:

- **What will you do in Math Today?** A free online resource of math activities (and more!) for young mathematicians, and their teachers and parents, interspersed with math + coding experiences. See www.researchideas.ca/wmt.

- **Math + Coding Stories:** Written by Alex Gadanidis & George Gadanidis. Join earthlings Pix & Vix in their Math + Coding adventures with Martians Zork & Gork. First two stories are freely available in HTML5 and in PDF at www.brainyday.ca/code.
- **Project updates:** Visit George Gadanidis' website, at www.joyofx.com, or follow on Twitter @joyofx.

Follow @joyofx

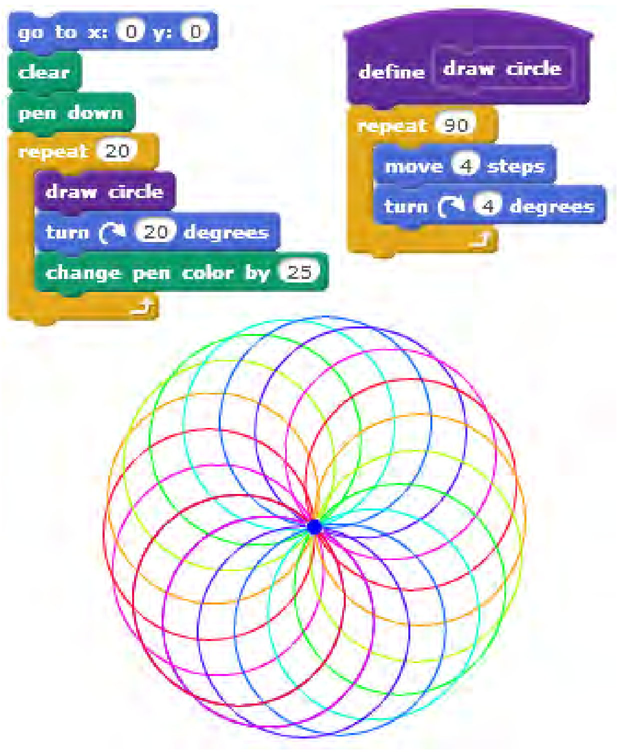
ABOUT THIS BOOK

This complete PDF version of *Coding for Young Mathematicians* is available under license from Western University (http://worldiscoveries.ca/technology/18155) as outlined below.

- **Individual/Teacher License:** An individual/teacher who has purchased a license from Western University may used it in their personal classroom teaching.
- **School License:** All teachers in a school for which a license has been purchased may used it in their personal classroom teaching.
- **School District License:** All teachers in a school district for which a license has been purchased may used it in their personal classroom teaching.

© 2015 George Gadanidis. All rights reserved.

TABLE OF CONTENTS



```
go to x: 0 y: 0
clear
pen down
repeat 20
    draw circle
    turn ↻ 20 degrees
    change pen color by 25
```

```
define draw circle
repeat 90
    move 4 steps
    turn ↻ 4 degrees
```

1. USING THIS BOOK

The math and coding activities in this book have been classroom tested in grades 1-8.

1.A. PICK-A-PATH

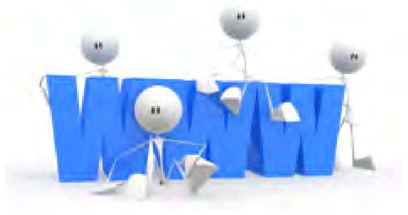
The book is designed to read and use in the order written OR you can skip to chapters that interest you. Chapters are self-contained and coding examples are easy-to-follow.

Each chapter contains hands-on math and coding examples to illustrate ideas.

Chapters 5-8 focus on specific math concepts. The math concepts and the coding examples in these chapters stretch across grades, and can be adapted for the specific grade(s) you teach.

1.B. USE-EDIT-CREATE

Throughout the book, links are provided to online resources, such as math + coding simulations and games, and samples of code that you and your students can use.



Use the code to see what is does.
Edit the code to create new variations, and better understand how it works.
Create your own code to explore new math relationships.

The **use-edit-create** cycle is a natural component of learning to code. It's also one that professional coders use. They **use** and study other people's code. They borrow and **edit** existing code. And they **create** their own code.

1.C. IT'S NOT ABOUT YOU AND ME

As a teacher, you might be apprehensive:

- "I've never coded before."
- "How can I teach something I haven't studied myself?"

But teaching is not about you and me. It's about the children in our classes, and their wonderful minds. Take a leap. Learn along with your students. You'll love it!

Here are Grades 7/8 student comments after a first experience with coding + math.

- *"I felt very curious in today's class because I had never used coding or even heard of it."*
- *"It was difficult at first but then I understood it."*
- *"It was very interesting and I got sucked right in."*

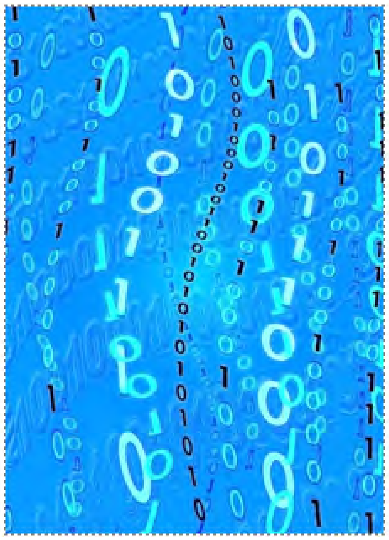
See also an interview at http://www.mathncode.com/interview1 with grades 1-3 and 7-8 teachers engaging their students with coding for the first time.

> *In my room (Grade 1) it was great to see the older kids (Grades 7/8) with the little kids and their little hands were just hovering above the keyboard and they they'd pull them down and your kids (Grades 7/8) were so beautiful at 'No you put your hands, you're going to do it.' And the celebrations that took place with the little successes and it was the energy as some of the kids developed really high level things and they would all run over and look and they would say 'Oh look at this' and they would say 'OK let's try that." And I ran to your room and see some wonderful and different things here and I ran back and said 'Wow, do you know what I saw in Ms. Silver's room?'*

2. THE CODING MOVEMENT

Digital technologies are part of our daily lives - we all use them but few people create them.

Coding (a.k.a. "computer programming") is the process used to create and control the apps, video games, and other digital artefacts around us.

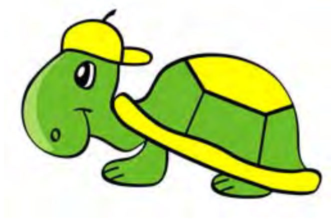


Proponents of coding suggest that young children should learn to code, to become producers as well as consumers of technology, to better understand and control their technological world, and to be better prepared to succeed in it.

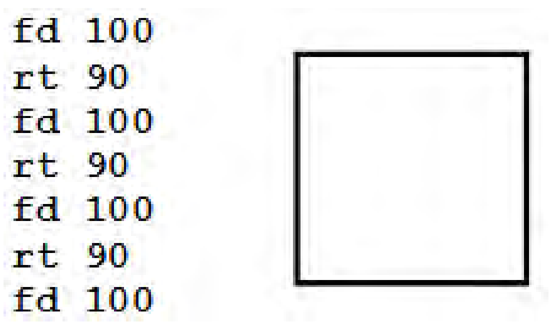
In this chapter we look at the past, present and future of coding in education, and its connection to mathematics education.

2.A. PAST

The most prominent coding movement of the past was based on Seymour Papert's **Logo**. In **Logo**, children use code to move a "turtle" on the screen, which can leave a trail of the path it travels.



For example, the **Logo** code below draws a square.



```
fd 100
rt 90
fd 100
rt 90
fd 100
rt 90
fd 100
```

We get the same result with more concise code using **Logo**'s **`repeat`** function:

```
repeat 4 [fd 100  rt 90]
```

NOTE
- **`fd 100`** means forward 100 pixels; and
- **`rt 90`** means turn right 90 degrees.

Papert developed **Logo** as a mathematics learning environment. He wrote in his 1980 book *Mindstorms: Children, Computers, and Powerful Ideas* that **Logo** "is to learning mathematics what living in France is to learning French." (p. 6)

Papert also developed **Logo** to challenge "beliefs about who can understand what and at what age." (p. 4) Papert was a student of Jean Piaget, and although he agreed with Piaget that children need to construct their own understanding of mathematics (rather than learning ready-made rules and procedures), he disagreed with Piaget's developmental stages. Papert suggested the stages identified by Piaget are not in children's minds but in the way mathematics is taught.

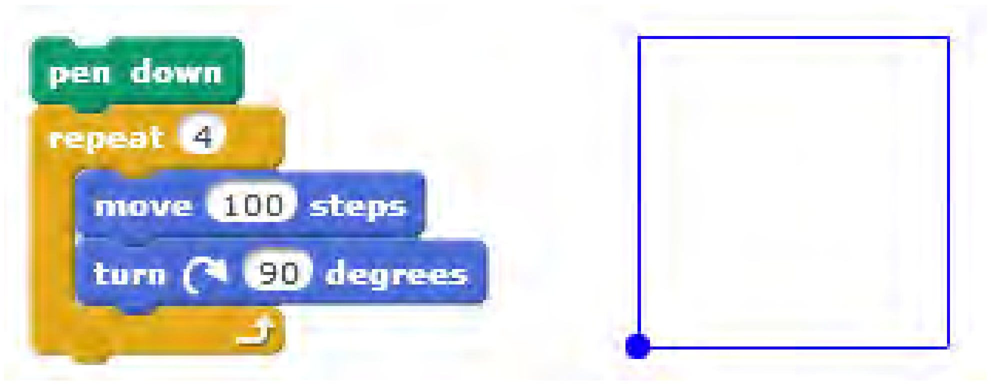
Logo was designed to set free children's developmental potential. **Logo** has a low floor, so children can engage with minimal prerequisite coding and math knowledge, and a high ceiling, so children have opportunities to extend their thinking to more complex concepts, representations and relationships.

Papert also developed a robotic "turtle" which children move using computer code. Like its screen-based counterpart, it had the ability to leave a trail of the path it travelled.

Papert also cooperated with Lego to develop Lego *Mindstorms*, a series of kits for creating programmable robots.

Papert's ideas continue to be very influential today, and current child-friendly coding environments, such as **Scratch**, contain **Logo**-like features.

For example, the **Scratch** code below draws a square on the screen.



```
pen down
repeat 4
  move 100 steps
  turn ↻ 90 degrees
```

TRY IT!
- Go to http://scratch.mit.edu/projects/31903528/#editor
- Change the values in the code and observe the effect:
  - Change **repeat** to 3 and **turn** to 120
  - Change **repeat** to 5 and **turn** to 72
  - Change **repeat** to 6 and **turn** to 600
  - Change **repeat** to 8 and **turn** to 45
  - Change **repeat** to 100 and **turn** to 145

Although **Logo** offered a lot of potential for engaging and educating children, and was implemented as a mathematics and coding experience in a numerous classrooms, it is fair to say that **Logo** did not gain widespread acceptance, and its underlying educational philosophy did not infiltrate mandated curriculum documents. An obstacle was that **Logo** was typically presented as an alternative to the existing math curriculum rather than its partner. To take up **Logo** you had to not only accept coding as a goal, but in many ways also give up on the mathematics curriculum you were already teaching. This made **Logo** difficult to implement.

2.B. PRESENT

In recent years, there have been renewed calls for young children to learn to code. Some of the calls are from universities, such as MIT and Carnegie Mellon, which have developed their own child-friendly coding languages (**`Scratch`** and **`Alice`**, respectively). Other calls have come from technology leaders, such as Bill Gates of Microsoft and Dick Costello of Twitter, from non-profit coding advocacy organizations, such as code.org, and from leaders in the field of computational thinking, such as Jeannette Wing (2006).

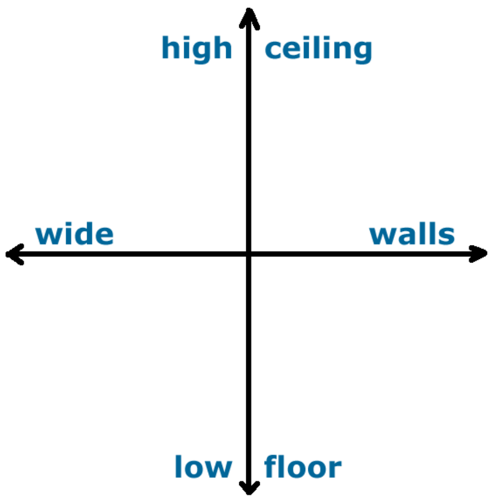
Unlike previous calls for coding in education, there is now more serious consideration given to including coding skills in curriculum. For example, starting in the Fall of 2014, the UK national curriculum mandates that children at all grades will learn to code. The new curriculum replaces the previous focus on "computer literacy" with a focus on "teaching children how to code, create programmes and understand how a computer works" (UK Government News Release, 4 February 2014).

Below are some of the Grades 1-6 coding expectations from the new UK curriculum (Berry, 2013, p. 6):

- Grades 1-2
  - understand what algorithms are
  - create and debug simple programs
  - use logical reasoning to predict the behaviour of simple programs
- Grades 3-6
  - design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems
  - use sequence, selection, and repetition in programs
  - work with variables and various forms of input and output

Like Papert's idea of a low floor and a high ceiling, these skills represent the minimum of what is to be taught, but not the maximum (Berry, 2013).

Resnick, a principal developer of the **`Scratch`** coding language, suggests a second dimension of "wide walls", which supports "many different types of projects so people with many different interests and learning styles can all become engaged" (Resnick et al, 2009, p. 63). "Wide walls" also allow and motivate children to engage with a wider audience, as opposed to just with their classroom peers (Gadanidis, 2014).



Attempts are being made to create similar low floor, high ceiling, wide walls learning with digital circuit making products such as *Arduino* and *LilyPad Arduino* (Kafai & Burke, 2014; Buechley, 2010; Buechley et al, 2013), where coding is used to control how digital "tangibles" function and how they respond to their environment.

Unlike **Logo**'s coupling of coding with mathematics learning, the current focus sees coding more as an end in itself, seeking to create a tech-savvy workforce for a 21st century economy. UK's Chancellor of the Exchequer noted:

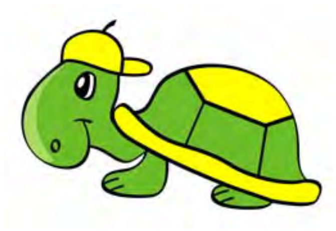
> We are already making Britain the place to start and grow a tech company. This year by introducing coding into the curriculum we are also making sure Britain is the place to learn to code. [...] In the 21st century, the ability to code and program a computer is no longer a nice-to-have, it's an essential. (UK Government News Release, 4 February 2014)

2.C. FUTURE

Will the coding movement gain strength or will it subside?

What can we learn from the past?

Logo-based coding had the advantage of being connected to an existing curriculum area, and was offered not only as a coding movement but also as mathematics education reform.



However, as already mentioned, **Logo** was typically presented as an alternative to the existing math curriculum rather than a partner. To take up **Logo** you had to in many ways give up on the mathematics you were already teaching. This made **Logo** difficult to implement.

Today, the focus on coding is offered as its own curriculum, and connected to digital literacy rather than a traditional subject area, like mathematics. This is the approach taken so far in the new UK coding curriculum.

The approach taken in this book is to show that there important ways of coupling coding with mathematics, while still covering the content in the existing mathematics curriculum.

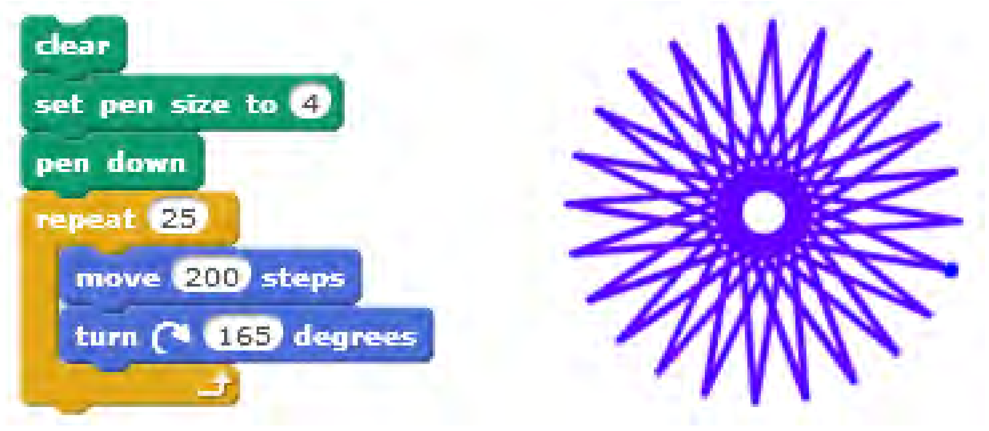
Some integration of mathematics and coding has benefits: (1) it helps address the issue of a crowded curriculum; (2) it provides a meaningful context for learning to code; (3) the modelling of mathematical concepts made possible through coding can enhance understanding; and (4) mathematics and coding share a similar logical structure, with a common focus on computational thinking (Gadanidis, 2014).

This does not mean that coding cannot or should not also be taught on its own. However, if coding continues to be offered solely as an end in itself, without links to the existing curriculum that teachers have to teach, it will be more difficult to implement effectively, and its future in an already crowded curriculum may be uncertain.

3. THE CODING ADVANTAGE

The modelling of math concepts made possible by coding offers three advantages:

1. It makes math concepts **tangible**.
2. It represents math concepts **dynamically**.
3. It puts **students in control**.



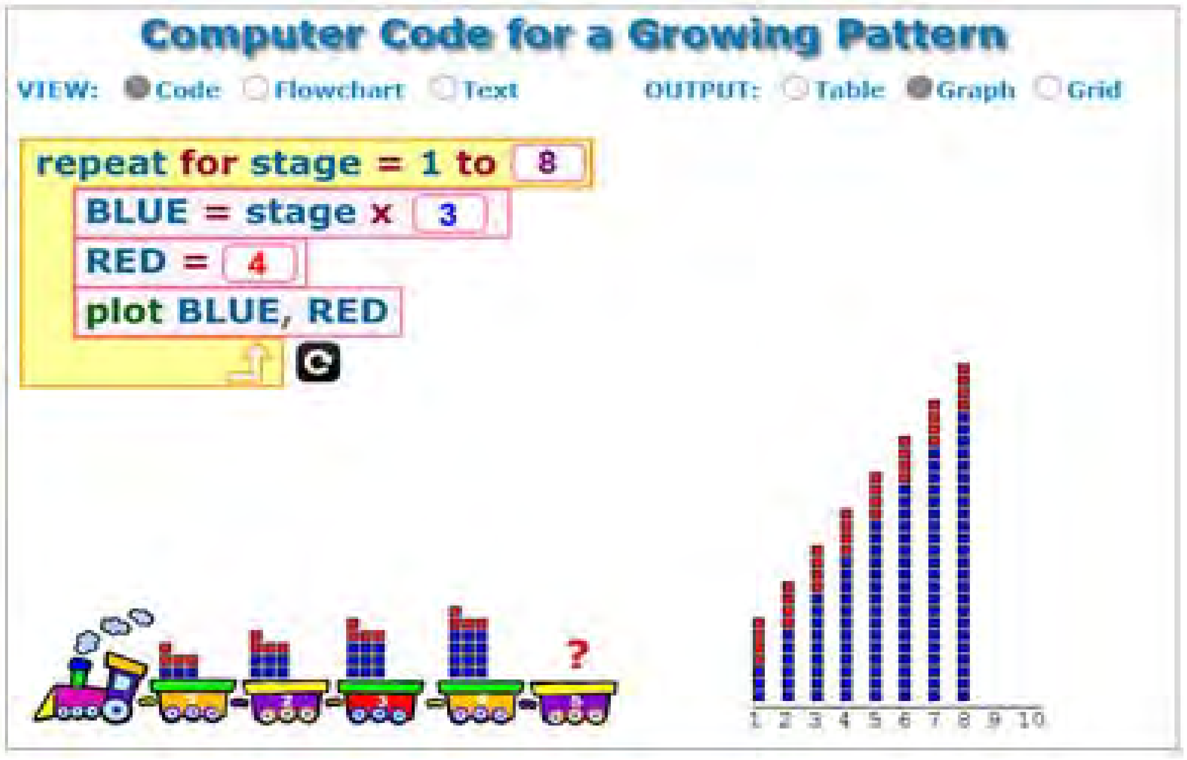
```
clear
set pen size to 4
pen down
repeat 25
    move 200 steps
    turn ↻ 165 degrees
```

Explore these advantages in the following pages, with coding + math examples from elementary school classrooms.

3.A. TANGIBLE

You might be thinking, "How could something digital, like coding, make math tangible?"

Below is an example where coding is used to help us to represent math patterns as (images of) objects on train cars and as bar graphs that we can manipulate.


Computer Code for a Growing Pattern
VIEW: Code Flowchart Text
OUTPUT: Table Graph Grid
?
1 2 3 4 5 6 7 8 9 10

```
repeat for stage = 1 to 8
    BLUE = stage x 3
    RED = 4
    plot BLUE, RED
```

TRY IT!

- Go to www.mathNcode.com/sims-growpatt.html
- Change the values in the code and notice the effect.

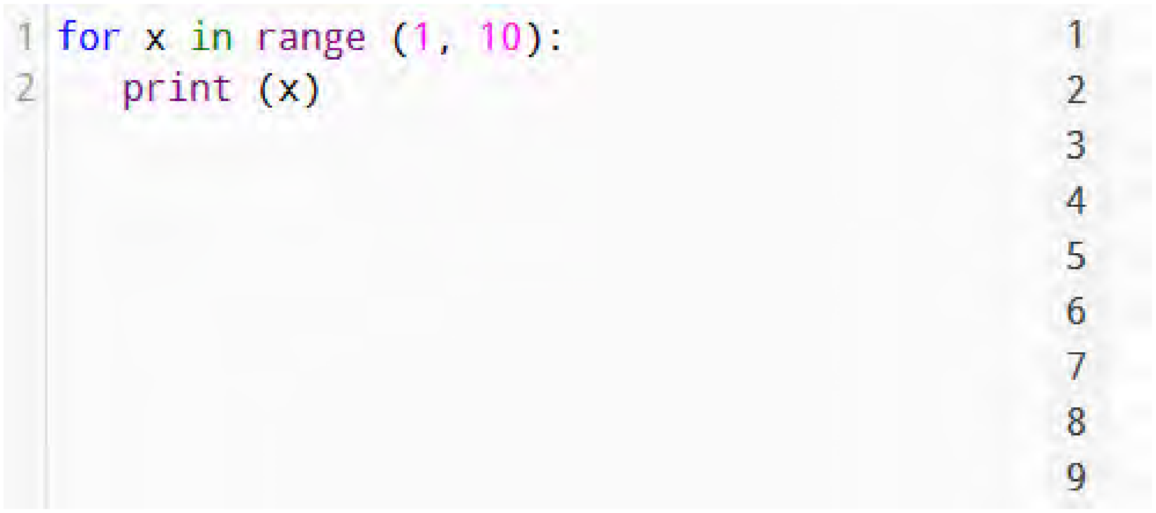
In the next example, young students working with natural, even and odd numbers, use the **Python** programming language to represent and manipulate them through code.

WHAT IS PYTHON?

- **Python** is a widely used computer coding language.
- It is designed to make computer code readable and easy-to-understand.
- It also allows for expressing concepts in fewer lines of code than other popular computer coding languages.

Natural Numbers. The two lines of **Python** code below list the natural numbers (**1, 2, 3** ...).

- The letter (or variable) **x** represents the natural numbers.



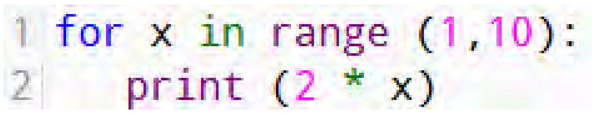
```
for x in range (1, 10):
    print (x)
```

```
1
2
3
4
5
6
7
8
9
```

NOTE
- Notice that `range (1,10)` only prints 9 natural numbers.
- **range** starts at **1** and goes up to, but not including, **10**.

TRY IT!
- Go to http://cscircles.cemc.uwaterloo.ca/console
- Type in the code from the above examples.
- Click on **Run** to see the **output**.

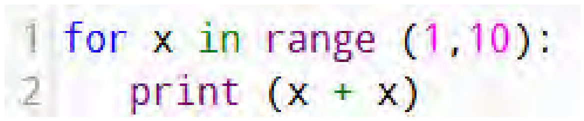
Even Numbers. This **Python** code lists the even numbers (**2, 4, 6** ...).



```
for x in range (1,10):
    print (2 * x)
```

Notice that we get the even numbers by doubling the natural numbers.
- The expression **2 * x** represents the even numbers.
- The asterisk ***** means multiply.

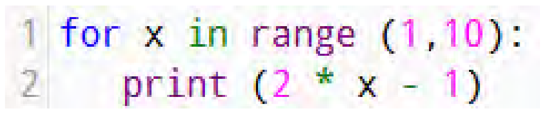
We could have also used the expression **x + x** to get the same result.



```
for x in range (1,10):
    print (x + x)
```

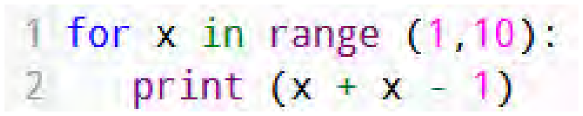
Odd Numbers. Here is the code for listing the odd numbers **(1, 3, 5** ...).

- The expression **2 * x - 1** subtracts **1** from the expression for the even numbers **2 * x**, to get the odd numbers.



```
for x in range (1,10):
    print (2 * x - 1)
```

The expression **x + x - 1** would also work.



```
for x in range (1,10):
    print (x + x - 1)
```

Coding makes expressions like **2 * x** feel "tangible" by turning them into code objects that can be manipulated, listed, printed, graphed, etc.

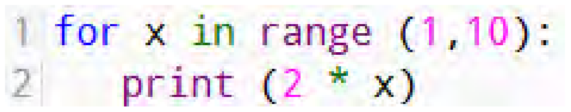
TRY IT!

- Go to http://cscircles.cemc.uwaterloo.ca/console
- Type in the code from the above examples.
- Click on **Run** to see the **output**.
- Try using different numbers and operations, and observe how the **output** changes.

3.B. DYNAMIC

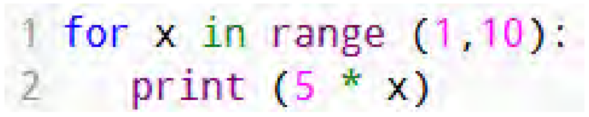
Coding can help represent math concepts dynamically.

For example, the **Python** code below prints even numbers.



```
for x in range (1,10):
    print (2 * x)
```

Changing **(2 * x)** to **(5 * x)** would list multiples of **5** rather than even numbers.



```
for x in range (1,10):
    print (5 * x)
```

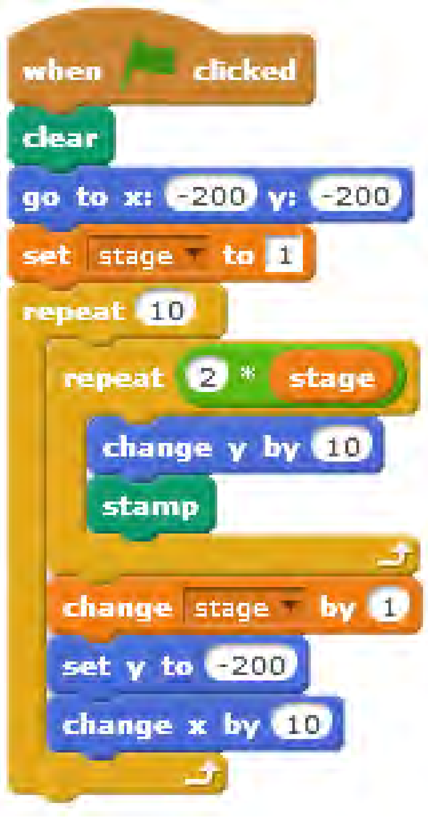
The dynamic relationship between the code and the math that is represented is enhanced because changing the code shows the change immediately.

Let's look at another example, using the **Scratch** coding language.

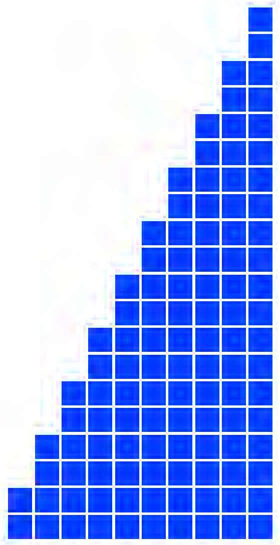
WHAT IS SCRATCH?

- **Scratch** is a child-friendly, drag-and-drop computer coding language.
- Students attach code to objects (sprites) to create interactive simulations, stories and games.

The **Scratch** code below plots the first 10 even numbers as a bar graph.



```
when [flag] clicked
clear
go to x: (-200) y: (-200)
set [stage] to (1)
repeat (10)
    repeat ((2) * (stage))
        change y by (10)
        stamp
    end
    change [stage] by (1)
    set y to (-200)
    change x by (10)
end
```



USE-EDIT-CREATE

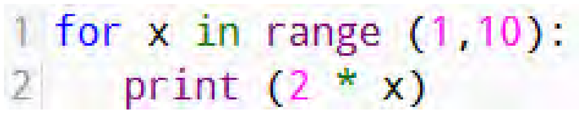
- Go to http://scratch.mit.edu/projects/30699054/#editor to see and play with the above code.
- Click on the green flag at `when [flag] clicked` to **Run** it.
- Change `((2) * (stage))` to `((3) * (stage))`.
- Click on the green flag to **Run** it and see the effect.
- Change `((2) * (stage))` to `(((2) * (stage)) - (1))` as follows:
    - o Click on the **Scripts** tab and then select **Operators**.
    - o Drag and drop `(( ) - ( ))` to replace `((2) * (stage))`.
    - o Drag and drop `((2) * (stage))` into the first blank of `(( ) - ( ))`, and type 1 in the second blank to get `(((2) * (stage)) - (1))`.
- Click on the code to **Run** it and see the odd numbers as a bar graph.

WHAT KIDS SAY ABOUT PYTHON AND SCRATCH

- At first glance, kids familiar with **`Python`** say that **`Scratch`** looks complicated.
- After they try **`Scratch`** for 15-20 minutes, they say it's easier and more fun than **`Python`**.
- What do you think?

3.C. STUDENTS IN CONTROL

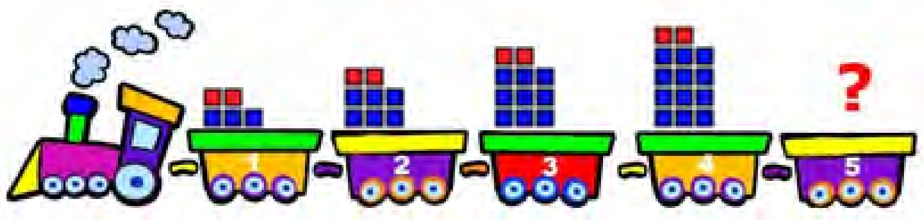
When students use an expression like **2 * x** in their code to represent a math concept (even numbers, in this case), they have control over the expression.



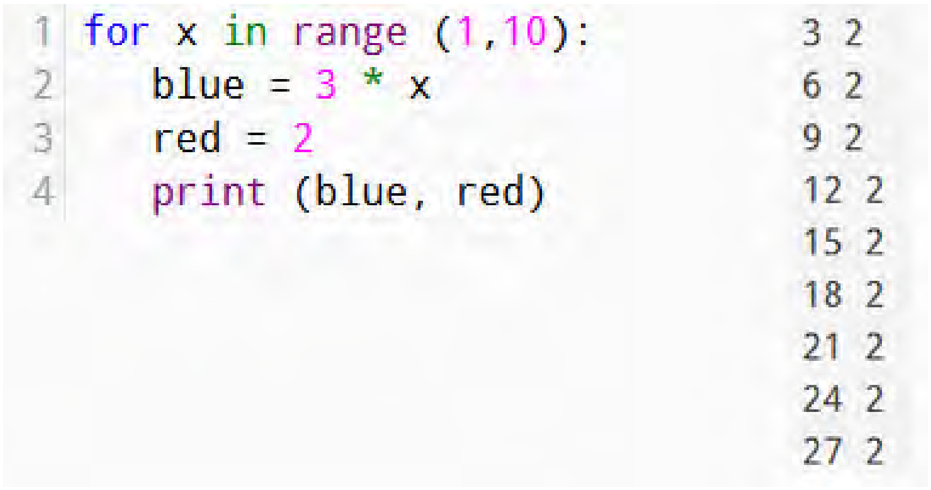
```
for x in range (1,10):
    print (2 * x)
```

They can easily change the expression to represent a different pattern and see the effect.

For example, below is a growing pattern where the **blue** blocks increase by **3** at each stage, and the **red** blocks that are always **2** in number.



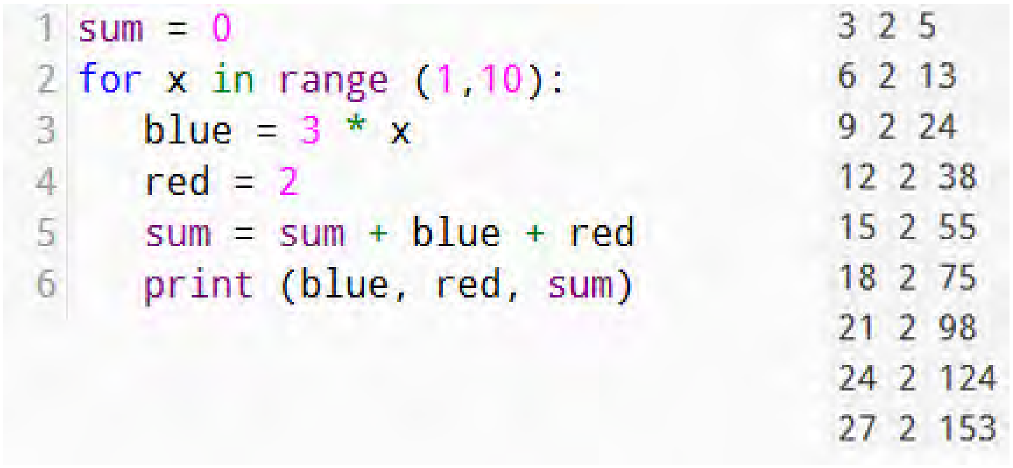
Below is the **Python** code that could model this pattern, and print the number of blue and red blocks at each stage (shown below-right).



```
for x in range (1,10):
    blue = 3 * x
    red = 2
    print (blue, red)
```

```
3 2
6 2
9 2
12 2
15 2
18 2
21 2
24 2
27 2
```

Students can edit the code to also explore different sums of the **`blue`** and **`red`** blocks, as shown below.



```
sum = 0
for x in range (1,10):
    blue = 3 * x
    red = 2
    sum = sum + blue + red
    print (blue, red, sum)
```

```
3 2 5
6 2 13
9 2 24
12 2 38
15 2 55
18 2 75
21 2 98
24 2 124
27 2 153
```

TRY IT!

- Go to http://cscircles.cemc.uwaterloo.ca/console
- Type in the code from the above examples.
- Edit the code to explore a different pattern.
- Click on **Run** to see the **output**.

See Chapter 8 for more explorations of such growing patterns.

4. PEDAGOGICAL PRINCIPLES

The math and coding activities in this book are based on three pedagogical principles:

1. Cover the **curriculum**.
2. **Surprise** students mathematically.
3. Share **good math stories**.

Over the past 10 years, I have spent an average of 50 days each year in elementary school classrooms, in Canada and in Brazil, collaborating with teachers to develop effective ways of engaging children with mathematics. The three principles listed above have been the starting point for this work.

In this chapter, I illustrate these principles with math examples from elementary school classrooms.

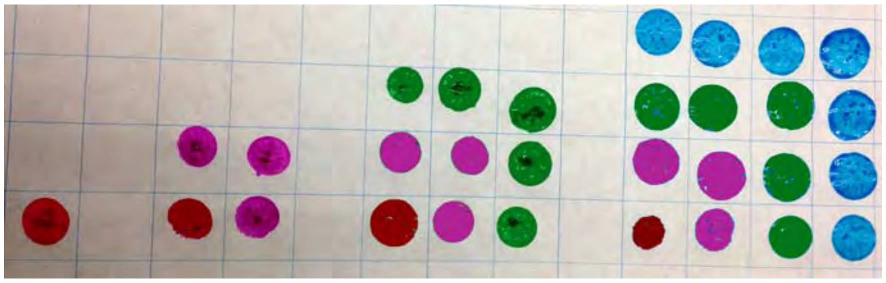
4.A. COVER THE CURRICULUM

Teachers are professionals and they take seriously their job of teaching the curriculum.

In all of the activities we design, we ensure that the content teachers have to teach is covered.

In the next section, you will be introduced to a grade 2 math activity that covers the following content:

- classifying numbers (odd and even)
- creating and extending growing patterns (1, 3, 5, 7 ...)
- using multiple representations of numbers and patterns (symbolic, concrete, visual)



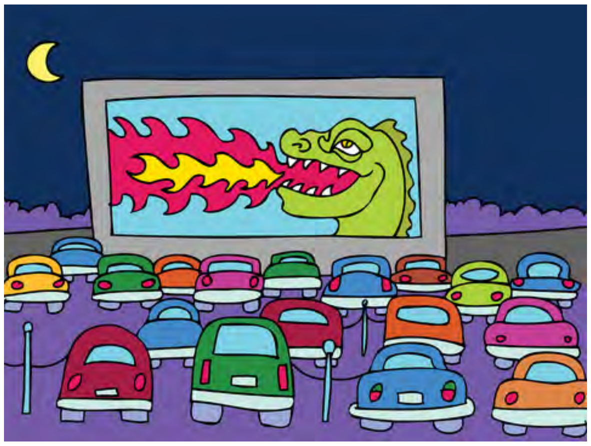
In addition, students have opportunities to learn that math can be surprising and interesting to talk about: "Odd numbers hide in squares!"

4.B. SURPRISE STUDENTS

To make surprise possible, we shift from "easy-to-learn math" to "math that makes students think hard".

"What's wrong with easy-to-learn math?" you might ask.

Imagine watching a movie. Typically, as you watch the scenes unfold, you try to guess what might happen next. If the movie is easy-to-learn, your guesses are correct and the movie becomes predictable and boring.



"Easy-to-learn" movies are difficult to watch. We take pleasure from movies when our predictions are incorrect, when we are surprised, when we have to think hard to make sense of unexpected plot twists.

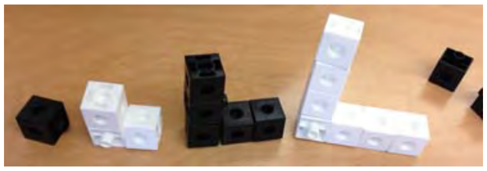
The same holds true for learning mathematics. Easy-to-learn math experiences don't hold our attention and they make learning difficult. Surprising math experiences draw our attention, offer us new insights, helps us enjoy thinking hard, and develop powerful conceptual understanding.

Watson and Mason (2007, p. 4) "see mathematics as an endless source of surprise, which excites us and motivates us. ... The challenge is to create conditions for learners so that they too will experience a surprise."

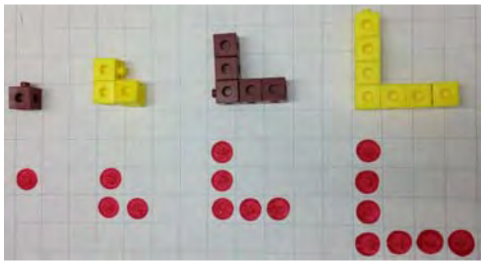
What does a math surprise look like?

Here is a math surprise from grade 2.

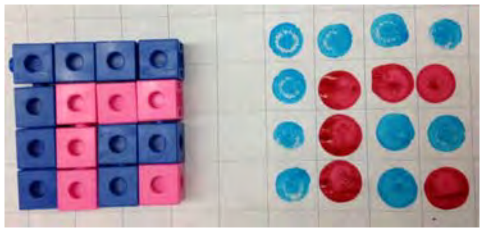
Children use linking cubes to create concrete representations of the odd numbers **1**, **3**, **5** and **7**.



They also use bingo dabbers to create matching visual representations.

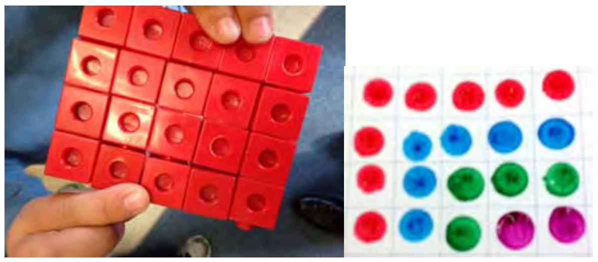


As students "play" with their representations, they discover that they fit together like spoons to form squares.



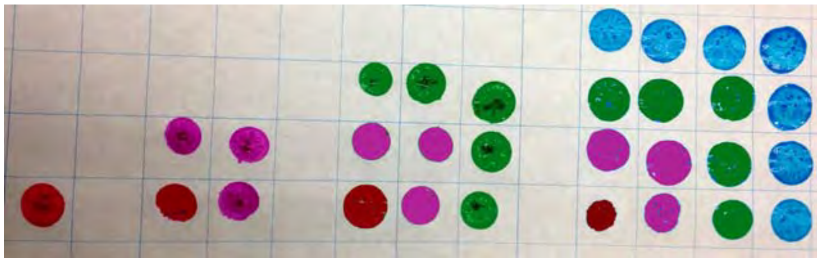
Odd numbers hide in squares? Now that's a cool math surprise!

Where do even numbers hide?



4.C. SHARE GOOD MATH STORIES

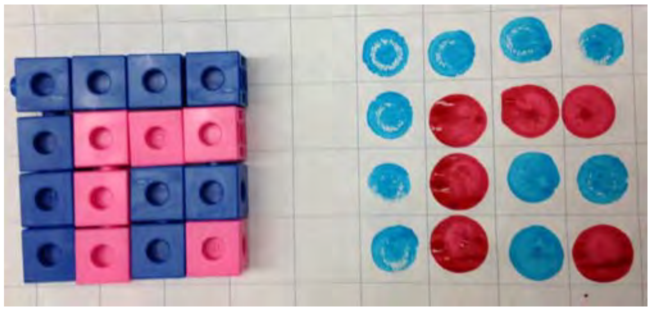
In the odd numbers activity, students learned that math can be surprising and interesting to talk about.



Students experienced a math surprise by looking at patterns where odd number **L**s fit together to form a square.

> The first **2** odd numbers make a **2 x 2** square.
> The first **3** odd numbers make a **3 x 3** square.
> The first **4** odd numbers make a **4 x 4** square.
> The first **10** odd numbers make a **10 x 10** square.
> The first **100** odd numbers make a **100 x 100** square.

The idea of fitting physical representations of odd numbers to form squares is actually a way of representing the sums of odd numbers.



In the image above we see that:

1 + 3 + 5 + 7

= 4 x 4

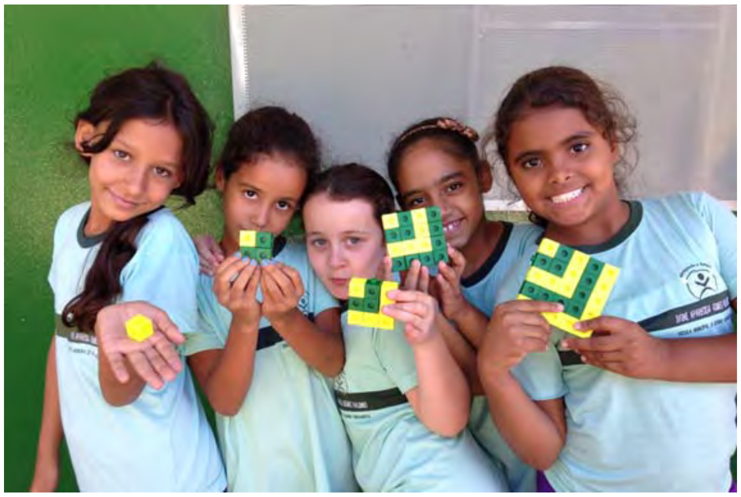
= 16

Students will encounter this idea again in grade 11, where they will use algebra to prove that the sum of the first **N** odd numbers is **N x N**.

You probably don't remember studying this topic in high school. It looked something like this:

$$\sum_{i=1}^{N} (2i - 1) = N \times N$$

You would have likely remembered this if you experienced it like these grade 4 students did in Brazil.



And if you experienced it this way, you would have been eager to share your learning with family and friends.

When math surprises you, it draws your attention, it makes you think hard, and it helps you learn and understand.

Math that surprises you also makes for a good math story to share, when asked "What did you do in math today?"

9. RESOURCES

CODING ENVIRONMENTS USED IN THIS BOOK

Scratch

- Website is at http://scratch.mit.edu
- Here you will find examples as well as a discussion forum
- Join Scratch (for free) so you can save your code

Python (University of Waterloo)

- Website is at http://cscircles.cemc.uwaterloo.ca/console
- Create an account (free) so you can save your code

OTHER CODING ENVIRONMENTS (partial list)

Alice

- Website is at http://www.alice.org
- A 3D programming environment that makes it easy to create an animation for telling a story, playing an interactive game, or a video to share on the web.

Android App Inventor

- Website is at http://appinventor.mit.edu
- A blocks-based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices.

MIT App Inventor

Blockly Games

- Website is at https://blockly-games.appspot.com
- Educational games that teach programming. Designed for children who have not had prior experience with computer programming.

Python (another version)

- Website is at https://www.python.org

10. REFERENCES

Aho, A.V. (2012). Computation and Computational thinking. *Computer Journal, 55*, 832-835.

Berry, M. (2013). *Computing in the National Curriculum. A Guide for Primary Teachers*. Bedford, UK: Computing at School. Retrieved 12 May 2014 from http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf

Buechley, L. (2010). Questioning Invisibility. *Computer, 43*(4), 84-86.

Buechley, L., Peppler, K., Eisenberg, M. & Kafai, Y. (2013). *Textile Messages: Dispatches From the World of E-Textiles and Education*. NY: Peter Lang.

Gadanidis, G. (2014). Young children, mathematics and coding: A low floor, high ceiling, wide walls learning environment. In D. Polly (Ed). *Cases on Technology Integration in Mathematics Education (pp. 312-344)*. Hersey, PA: IGI Global.

Kafai, Y. B. & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming.* Cambridge, MA: The MIT Press.

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books.

Resnick, M. et al (2009). "Digital fluency" should mean designing, creating, and remixing, not just browsing, chatting, and interacting. *Communications of the ACM, 52*(11), 60-67.

UK Government News Release (4 February 2014). Year of Code and £500,000 fund to inspire future tech experts launched. Retrieved 12 May 2014 from https://www.gov.uk/government/news/year-of-code-and-500000-fund-to-inspire-future-tech-experts-launched

Watson, A. & Mason, J. (2007). Surprise and inspiration. *Mathematics Teaching*, 200, 4-7.

Wing, J. M. (2006). Computational thinking and thinking about computing. *Communications of the ACM., 49*, 33-35.