Kyle Pearce asked me for some advice for GECDSB teachers concerned about the addition of Coding to the new Ontario Mathematics Curriculum. Here is what I wrote to him:

First, I would tell teachers Coding in 2020 is not your Mother’s coding any more! The proliferation of Block Coding has brought coding to a wider audience of people of all ages and all walks of life. While your Mother needed to worry about using the correct comma or colon or type of bracket in each line of code, coders in 2020 use blocks with the correct information embedded to build actions. A block is an icon that looks like a brick with either symbols or words on it to describe the action a block does. Blocks can be connected together to make a train or stack of multiple actions. So, if I want a robot to move on the floor of my classroom or a picture of a cat to walk across my device’s screen, I just need to input the distance as a variable in a block rather than write multiple lines of code with the correct symbols and it moves (and yes, even with young children we use the word variables!). This example highlights the second thing teachers should know about coding. They, and their students, can code in the real world and have robots or other items “do” things they can see, hear, and touch or they can code in a virtual world where the actions of the code appear on a screen. It is the virtual aspect of coding that is free and available to anyone with access to a device. There are activities that illustrate coding that do not require any technology and they are usually called Unplugged. These avenues, unplugged, virtual, and real world mean teachers and students can learn in a variety of environments. Access to learning to code should not be seen as having roadblocks.

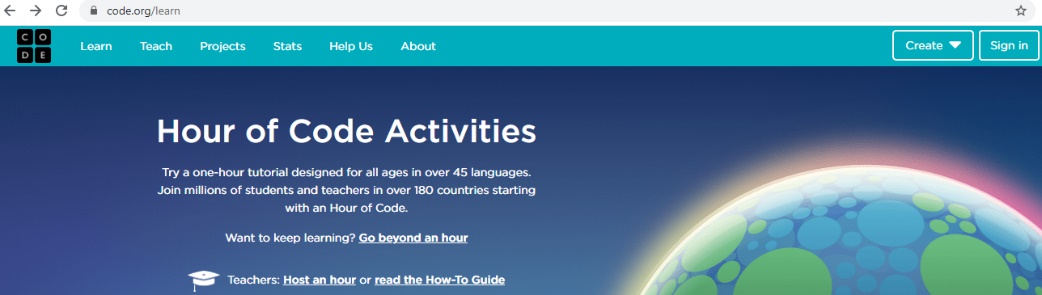
Second, teachers should know they can learn to code before, during, and after they teach coding to their students. They do not need to be a Level 4, A+ coder to teach coding. Part of the reason is that teachers are very good at breaking tasks down to individual steps and planning.

I like to use the analogy of making a sandwich. I know to get a slice of bread and put butter on top of the bread and then mayo on top of the butter. Next, I put on meat and lettuce and pickles. I add another piece of bread that has butter on top but I have flipped it over and the buttered side is facing down. I know that if I need to make a second sandwich, I can follow the same steps exactly or I can make some changes (variables, there’s that idea again!) to the recipe and I can make another sandwich. Once I am a sandwich maker, I can use the information I learned and transfer it to make/cook other things. It is the same with coding. Once I learn to make a square on grid paper, or give commands to have a friend move around in the gym (unplugged), I can make a virtual bee move around a rock to get to flowers or a virtual cartoon character to get to a treasure. Once I know how to make things move in a virtual world, I can use the same or similar blocks to have a BlueBot or DASH (or other robots) move around in the real world. Once I can move robots, I can use items such as Makey Makeys or Microbits to make objects other than robots move (or other actions I want them to do!).

Third, one of the best ways for teachers to learn to code is to **PLAY** with coding. Going on-line to try out apps or coding sites or turning on a robot will be the way their students learn. “What happens if I use this Block?” or “What happens if I use this number as the variable?” or “How can I make the robot run around in circles and beep?” will be the way children use coding. They may be testing out their hypotheses and learning by trial-and-error but to them it is just playing around. Teachers will say, and rightly so, they don’t have time to learn by just playing around, the stakes are too high, especially with a new Mathematics Curriculum to learn and implement and assess. So, a combination of free exploration playing and lesson plans for teachers will help them learn to code. Lessons are already out there and more are being created right now.

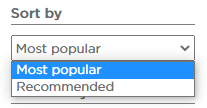
There are many places to start to learn. The place I suggest they start at is called Code.org ( <https://code.org/> ) 

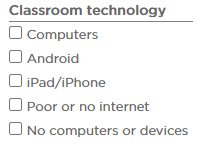
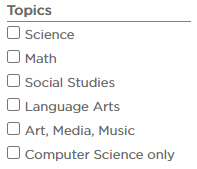
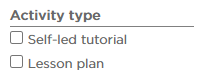
This site continues to grow and add more and more activities. Teachers might have heard of their Hour of Code Day but not realize that the site is available to use at any time.

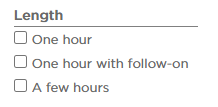
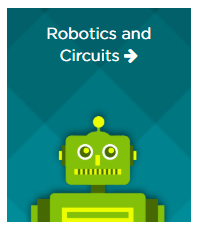


The ability to search activities by age, device, subject, time, and knowledge level gives a wide variety of choices to start and continue learning.



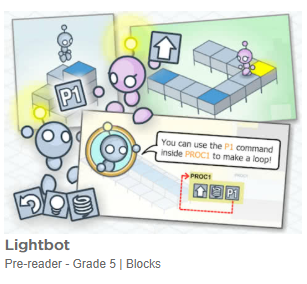
 and …

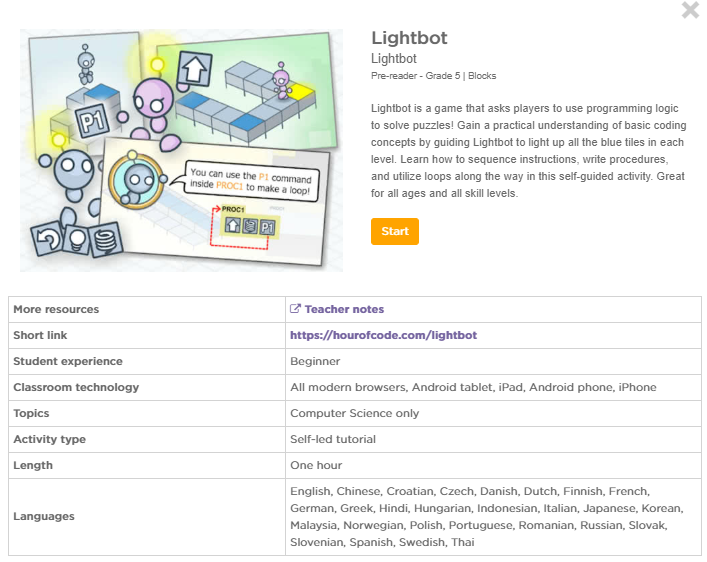
  

… enable teachers to find a place to start that suits their needs.

Clicking on an activity’s icon

 gives a brief outline of the app



I suggest looking at some of the following:

|  |  |
| --- | --- |
|  | Teacher Information: <https://code.org/hourofcode/frozen>  Activity: <https://studio.code.org/s/frozen/stage/1/puzzle/1> |
|  | Teacher Information: <https://code.org/hourofcode/hourofcode>  Activity: <https://studio.code.org/hoc/1> |
|  | Teacher Information:  <https://lightbot.com/resources.html>  Activity:  <https://lightbot.com/hour-of-code.html> |
| Code with Anna & Elsa and Angry Birds let students progress through a number of levels until they can make their own project.  Depending on the student, they may not make the leap to independent coding at the end of these activities.  They are coding with gamification, in other words, getting to the next level is the goal rather than learning and experimenting finding out “what happens if” - they are great to build interest but not necessarily to build knowledge.  The following ScratchJr, Scratch, and Google (CSFirst) activities are where learning will happen.  Lesson plans using these programs can be written to meet curriculum expectations for coding and mathematics as well as other subjects. | |

|  |  |
| --- | --- |
| These are the programs that I believe will be used frequently because they are free and are easily adaptable to multiple grades and differentiated instruction. They are low floor, wide walls, and high ceiling applications. | |
|  | Teacher Information:  <https://www.scratchjr.org/hoc.html>  Activity: (iPad)  <https://www.scratchjr.org/> |
|  | Teacher Information:  <https://resources.scratch.mit.edu/www/guides/en/MusicGuide.pdf>  Activity:  Click on Tutorials  Site: <https://scratch.mit.edu/> |
|  | Teacher Information:  <https://csfirst.withgoogle.com/c/cs-first/en/create-your-own-google-logo/overview.html>  Activity: <https://csfirst.withgoogle.com/c/cs-first/en/create-your-own-google-logo/create-your-own-google-logo/create-your-own-google-logo.html>  Site: <https://csfirst.withgoogle.com/s/en/home> |

GECDSB has kits for robots: BlueBot, DASH, Lego WeDo kits, and Lego Mindstorms kits.