THE INCLUSIVE TECHNOLOGY CLASSROOM

1. Ensure you are familiar with the tool you are teaching but don't worry about not being the 'expert' — allow your students to teach one another!
2. Have a growth mindset and use tech failures as an opportunity for learning. It's a great example of 'debugging!'
3. It's all about collaboration! Employ subject integration, group work, and peer-to-peer mentoring.
4. Bring outside experts in. Invite guest speakers, co-teachers and volunteers from the community to lead mini lessons and be there as extra coding support.
5. Maintain a diverse mentorship presence (e.g. recruiting volunteers or inviting women in the industry as guest speakers and mentors).
6. Encourage your learners to ask others for help first, before coming to you. (use the 'ask 3 before me' protocol)
7. Looping back lessons to why coding matters — not just how to code. Connect coding and technology to meaningful career paths (not just programming) and learner interests.
8. Keeping biases in check is important. We often unintentionally guide boys towards 'boy' things and girls toward 'girl' things. Let the students and their creativity be your guide — what do each of them want to explore and learn?
9. Be aware of imposter syndrome and celebrate accomplishments!
10. Consider creating a troubleshooting checklist with your learners.
11. Be creative, don't be afraid to fail and most importantly, have fun!

teachers learning code

Funded by: Canada

CODING VOCABULARY

Algorithm: a step-by-step set of operations to be performed to help solve a problem

Array: a special variable that can store more than one value at a time; items are ordered by a number so that we can access them later

Boolean Logic: 'and', 'or', 'not' are examples of boolean operators; the values you are working with must be either true or false

Conditionals: making decisions based on conditions (ie. if it is raining, then open your umbrella)

Debugging: finding problems in code and solving them

Events: one thing causing another thing to happen (ie. 'when green flag is clicked' block in Scratch)

Function: a named section of a program that performs a specific task; there are often canned functions that exist already like the 'If on edge, bounce' block in Scratch; these are sets of instructions that can be used over again

Loops: running the same sequence multiple times (ie. 'repeat' or 'forever' blocks in Scratch)

Modularizing: exploring connections between the whole and the parts; breaking down a project into smaller chunks of code

Operators: mathematical and logical expressions (ie. 'X + X' block in Scratch)

Parallelism: making things happen at the same time

Remixing: taking an existing project or idea and making it new by changing or adding to it

Sequence: identifying a series of steps necessary to complete a task; computers read and perform commands in order from top to bottom

Syntax: the spelling or grammar of a programming language; the blockly structure in languages like Scratch removes the need for syntax

Variable: stores a piece of information that changes over time (e.g. score)