teachers learning code

HOW-TO GUIDE FOR BEGINNERS

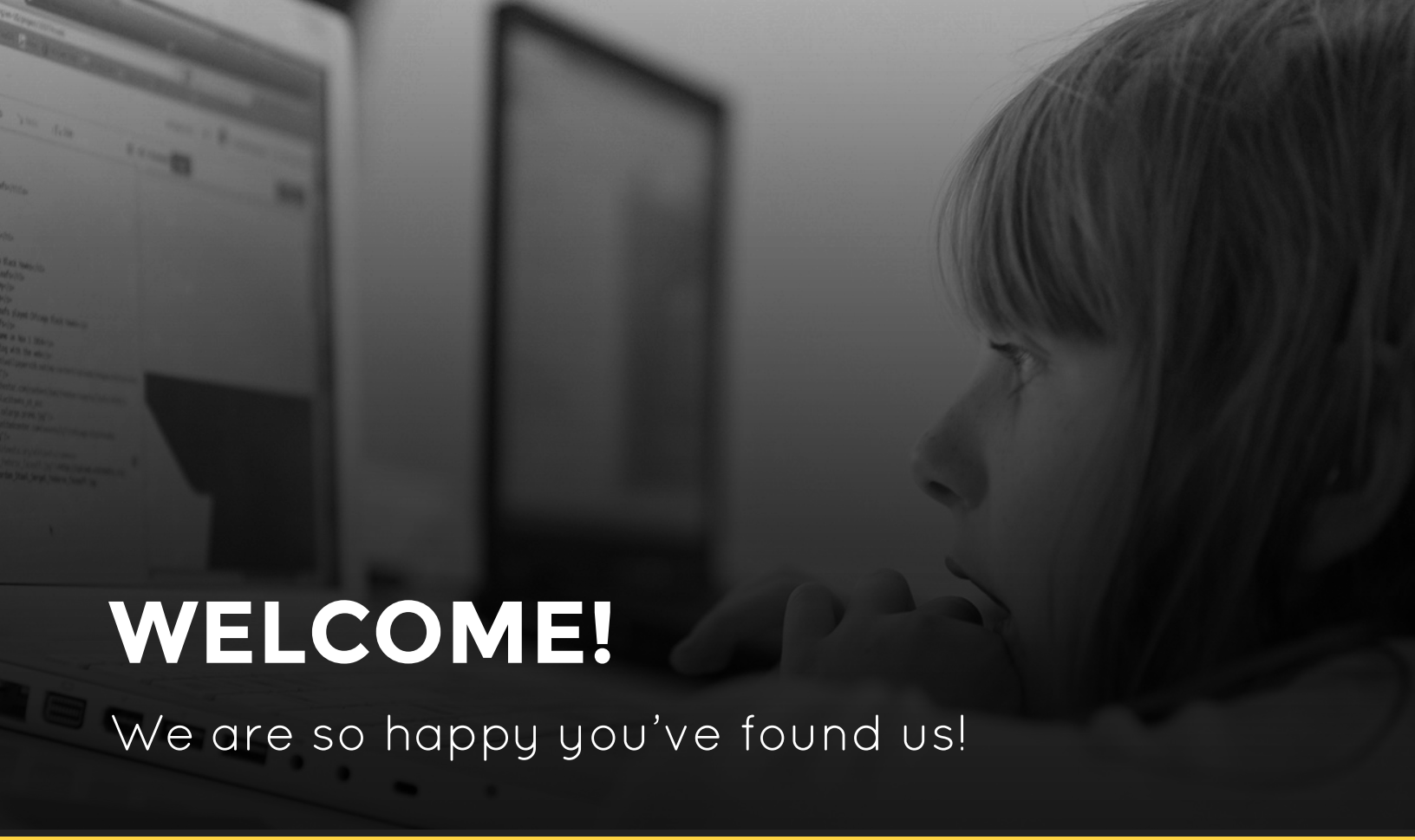
Start a coding club or introduce coding in your classroom.

TABLE OF CONTENTS

teachers learning code

is proudly supported by:

Scotiabank



WELCOME!

We are so happy you've found us!

Whether you are a teacher in a classroom, a program coordinator at a community centre, a home-schooling parent or a Girl Guide troop leader - we've put together this guide to help you teach kids to code.

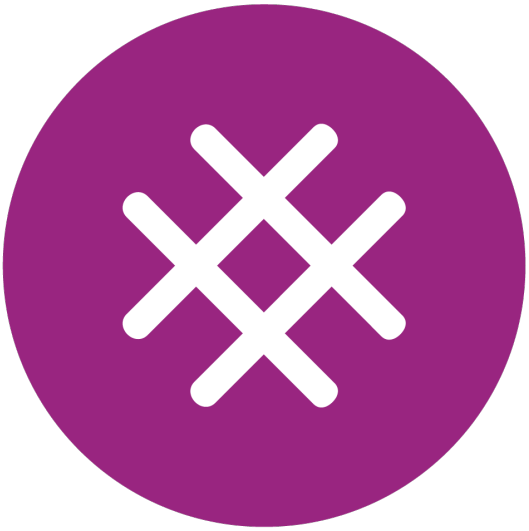
After running programs for several years, we have learned a thing or two. In this guide, we share many of our tips and tricks to get started, resources to familiarize yourself with code, plus lots of easy-to-follow and even easier to implement coding activities to empower and teach the future generation of technologists across Canada.

This guide is your jumping off point - where you choose to take your program is up to you!

We encourage you to use as much or as little of the guide as you like, to remix the challenges we've created or create new ones and share them with our community. Technology is creative - have fun!

WHAT IS LADIES LEARNING CODE?

DIGITAL LITERACY FOR WOMEN AND YOUTH.



We are a not-for-profit organization with the mission to become the leading resource for women and youth to become passionate builders - not just consumers - of technology by learning technical skills in a hands-on, social, and collaborative way.

Founded in 2011, we are a Canadian not-for-profit with the goal to teach 200,000 Canadian women and youth to code by 2020.

17,000+
adult learners

6,000+
youth learners

3,000+
mentors + volunteers

22+
city chapters

250,000+
hours of coding taught

To date, we've taught over 25,000 women and youth in one of our programs across the country. It is exciting and heartening to know that Canadians of all backgrounds are getting a first taste of coding at our workshops, and leaving with a greater sense of self-confidence. That's what digital literacy is all about! It's this community that we can help grow and foster with your support.

www.ladieslearningcode.com

WHAT IS TEACHERS LEARNING CODE?

USING TECHNOLOGY TO CHANGE THE WORLD THROUGH TEAMWORK, CREATIVITY, AND, OF COURSE, CODE.

Teachers Learning Code is a program designed by Ladies Learning Code and proudly supported by Scotiabank which is focused on inspiring kids to be builders –not just consumers - of technology through coding activities and challenges.

Our programming can be facilitated by non-technical or technical educators and youth program managers within schools and existing community groups. Teachers Learning Code is an approach to introducing coding that can be scaled back or expanded depending on the needs of your group of learners.

This 'how-to' guide and our coding challenges are designed for students in grades 3-8 but we've had younger and older students use them as well. They were all created with the core elements and features that our youth programs are known and loved for.

OUR FORMULA FOR SUCCESSFUL PROGRAMMING

Our content has been designed specifically with girls in mind and creating gender-balanced tech environments

Research has showed us that girls approach the computer as a "tool" useful primarily for what it can do; boys more often view the computer as a "toy" or an extension of the self. At Ladies Learning Code, we don't only teach "how" to build, but also "why" it is important. We offer an opportunity for girls and boys to learn about technology and collaborate together.

Fun and engaging content

We **LOVE** technology and want to pass it on. With the help of developers, designers and professional educators, we've created content that not only showcases beginner-friendly concepts, but also ideas to be excited about.

The opportunity to learn and collaborate alongside like-minded students

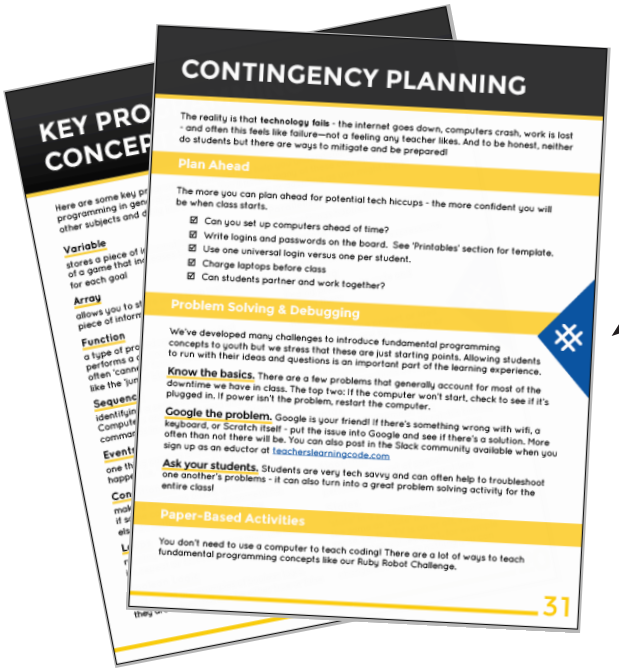
Through hands-on team-based exercises kids will feel part of a resourceful group who are all curious about technology, and find learning about it both intriguing and rewarding.

We're all about confidence and empowerment

Teachers Learning Code activities and challenges are all about instilling the growth mindset in learners. We're all about teaching girls and boys how to learn from their mistakes and developing confidence and empowerment through learning and building.

SOME WAYS TO INTRODUCE CODING IN THE CLASSROOM

- 10 or 12-week lunchtime club
  (each session is 45 minutes to 1 hour in length)
- 8 or 10 week after school program
  (each session is 1.5 to 2 hours in length)
- Standalone activities in tech class
- Integrated into any class like math, science or art



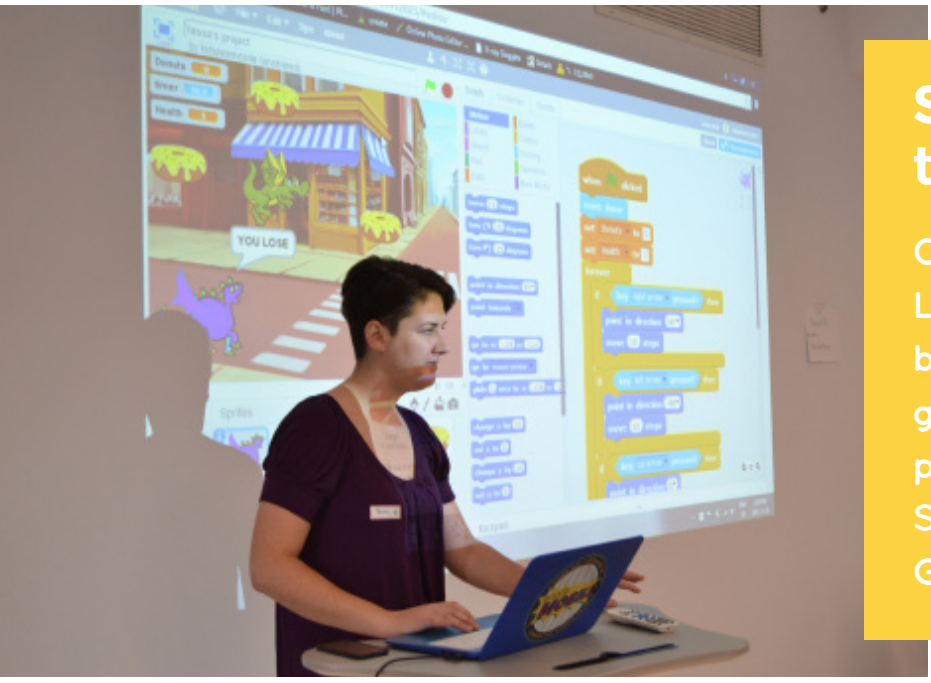
TIP:

Throughout this guide, we've tabbed sections that contain key concepts or resources you can easily refer back to.

HOW TEACHERS LEARNING CODE WAS DEVELOPED

At Ladies Learning Code, we've spent the last four years teaching thousands of youth across the country through workshops, camps and after-school programs through Girls Learning Code and Kids Learning Code. Our content has been developed in partnership with educators and industry-leading experts and tested, time and time again, with girls and boys between the ages of 6-13 inside and outside of the classroom.

For **Teachers Learning Code** specifically, we took this methodology and curriculum development process one step further and engaged non-technical educators in the planning, development and execution of the material. We hosted focus groups, sent out surveys, co-taught with teachers in schools and community centres, observed our materials being used in the classroom and iterated, iterated and iterated again -- all with the ultimate goal of developing a program that works for students and educators alike.



Special thank you to our Partners!

Our programs, like Teachers Learning Code, would not be possible without the generous contributions of our partners like: TELUS, Microsoft, Scotiabank, Georgian Partners, Google, Autodesk and more!

WHY TEACH STEM?

Technology is everywhere and it's not going anywhere. Science, technology, engineering and math (STEM), especially when integrated with other disciplines, are the skills of the future. We want to equip Canadian youth with the critical skills they need to navigate the world we live in today and thrive in the future.

But why teach coding?

According to Code.org, over the next 10 years there will be a shortage of close to one million developers in North America. Learning to code can lead to rewarding and lucrative careers for our youth. But, teaching kids to code is about more than just helping children understand the technology they are using and secure employment in the future. At a fundamental level, it improves problem-solving and thinking skills.

We think it's important for youth to learn to code for a few reasons:

1. **Coding is a superpower.** Learning to code let's kids build -- not just consume the technology around them like video games, websites, robots and more.
2. **Coding helps kids develop new ways of thinking.** Learning to code lets kids build -- not just consume the technology around them like video games, websites, robots and more.
3. **Coding helps kids understand the world around them better.** If we teach biology and mathematics in order to understand the world around them, then knowing the basics of how computers communicate and how to engage with them should be a given.
4. **Coding can help change the world.** Empower kids to use technology as a creative tool to build solutions for problems or challenges people face everyday.
5. **Coding is fun!** We want kids to experience the satisfaction and thrill of building something of their very own.

LEARNING OBJECTIVES

Coding is really about how you solve a problem, rather than learning a specific language or tool like Java or Scratch. Programming languages evolve and change all of the time but at the core, the fundamental blocks of how you think don't change. Computational thinking, or process for solving problems, can be taught even without learning a specific programming language. The code is just the tool that facilitates solving a challenge in a particular way. This is important to remember as educators - you don't need to know all the syntax of a language to teach it - you just need to understand the logic of solving problems - logic that you've already likely developed in your journey as an educator. Now, it's time to apply that logic to teaching code.

By teaching kids to code, we're teaching:

Computational Thinking

- Logical reasoning
- Critical thinking
- Pattern recognition
- Solving complex problems by breaking them down into simpler parts
- Debugging problems
- Developing ideas from initial concepts to a final project

Creativity & Collaboration

- Design thinking
- Innovation

Concepts About Computers

- Computer programs are created by humans and they tell the computer exactly what to do
- Computers aren't that bright or intuitive - they don't understand things the way humans do. You need to be exact and precise with your instructions to computers and instruct them step-by-step
- You don't have to be an expert to write code - you just need clear and careful thinking

LEARNING OBJECTIVES

Digital Citizenship

- Establishing a positive attitude towards building not just consuming technology
- Empowering kids to 'look under the hood' and ask questions about the technology they consume

Fundamental Programming Concepts

There are hundreds of computer programming languages out there and although they may look nothing alike to the human eye at their core, they are all the same. There are fundamental concepts and ways to interact with a computer. We'll be using Scratch as a tool to help us teach kids these concepts in a fun, relevant way.

- Variables
- Data
- Events
- Sequencing
- Iteration and loops
- Conditional statements
- Functions
- Parallel execution
- Boolean logic
- Random numbers
- Debugging
- Integrated math topics

COMPUTATIONAL THINKING

Computational thinking is the cornerstone to coding but, what exactly is it? Computational thinking is about looking at a problem in a way that a computer can help us to solve it and then we'll use technical skills (like a coding language) to tell the computer what to do to solve it. Computational thinking concepts aren't unique to coding - you'll notice they are principles many people use in their day to day lives to solve problems of all sorts. Computational thinking involves these key steps:

Logical reasoning

Computers aren't intuitive - they are predictable. Logical reasoning allows kids to work out what a computer will do. This process comes very quickly to kids as they develop early on a mental model for what technology does.

Algorithms

Algorithms are a set of instructions for the computer to do something. Algorithms are common in our everyday lives - a lesson plan is an algorithm for a class or a recipe for making our favourite dish. Writing out step-by-step instructions in plain English is what we call 'pseudo code'.

Decomposition

Code is complex and problems are complex. An important part of computational thinking and coding is breaking down problems into smaller more manageable steps like we might break down a book report into different sections.

Abstraction

After we break problems into smaller parts, abstraction helps us decide what's important and what's not. It helps manage complexity like a math problem where we decide what information we need to help solve an equation.

Patterns and generalizations

Patterning is an important part of learning to code. It helps us make predictions, create rules and solve more general problems. For example, a formula in math is a generalization that helps us solve many different problems.

HOW TO START

Want to start teaching code to your students?

ARE YOU:

STARTING AN AFTER SCHOOL PROGRAM OR CODE CLUB

You'll Need:

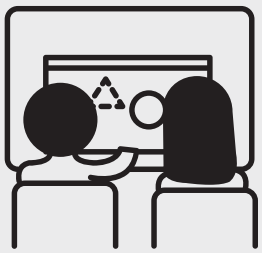
- ☑ **An Educator**
  To supervise students and facilitate coding challenges
- ☑ **Volunteer Mentors**
  Optional, but highly recommended. More on recruiting and training mentors under the 'Collaborative Teaching' section.
- ☑ **Learners**
  We've included a poster in the 'Printables' section you can use to help promote!
- ☑ **Venue**
  You'll need somewhere to meet
- ☑ **Access to Hardware**
  • Laptop or desktop computers
  • Projector + Projector Screen
  (or another way to display your screen to students)

INTEGRATING CODING INTO AN EXISTING CLASS OR PROGRAM

You'll Need:

- ☑ **An Educator**
  To supervise students and facilitate coding challenges
- ☑ **Volunteer Mentors**
  Optional, but highly recommended. More on recruiting and training mentors under the 'Collaborative Teaching' section.
- ☑ **Access to Hardware**
  • Laptop or desktop computers
  • Projector + Projector Screen
  (or another way to display your screen to students)

TOP TIPS FOR INTRODUCING STEM



Ensure you are familiar with the tool but don't worry about not being the 'expert' - allow your students to teach one another!

Have a clear vision for what you want to accomplish and find a champion in your school to co-teach with you - it's all about integrated learning!

Be creative, don't be afraid to fail and most importantly, have fun!

Have a growth mindset and use tech failures as an opportunity for learning. It's a great example for the popular developer concept of 'debugging'!

Bring outside experts in. Invite guest speakers and volunteers from the community to lead mini lessons and be there as extra coding support.

Let the students and their creativity be your guide - what do they want to explore more and learn?

COLLABORATIVE TEACHING

One of the biggest reasons our programs have become so popular across the country is because of a group of remarkable people we call mentors.

Mentors are volunteers who support our programs and help create a social and collaborative approach to learning. Whether you're technical or not, we highly recommend taking a collaborative teaching approach to introducing coding to your classroom. Whether you engage another teacher to co-lead your first coding class with you, engage a classroom of older students to mentor in your class or find external volunteers, mentors really make all of the difference for the students and for you as an educator to have that little bit of extra support.

Recruiting Mentors

The Role

While our Lead Instructors (educators) are front and centre leading, volunteer mentors are scattered throughout the room working with a group of learners. Mentors really are the magic. Mentors work with learners collaboratively which allows us to create a social and rewarding educational experience for everyone.

What it means to be a mentor

- You are either technical or have a passion for technology and learning
- You want to learn alongside the kids and help them with their projects
- You are prepared to assist kids in nding the answers to their questions
- You encourage an open, exible learning environment
- You maintain an optimistic view of technology
- You empower kids to become confident, inspired creators

Benefits of mentoring

- Experience that "warm and fuzzy" feeling!
- Network with other mentors
- Learn a new tool or language for your own personal benefit
- Develop leadership and communication skills
- Gain teaching experience
- "Give back" to the community

COLLABORATIVE TEACHING

We recommend finding mentors for your classes a few ways:

- Reach out to an experienced classroom teacher or local highschool or college/ university to find students who can mentor
- Connect with any existing volunteers you have for other programs and encourage them to review the material and mentor
- Reach out to local technology companies

You can find a poster in the 'Printables' section that you can put up or email to help you recruit volunteers.

Screening and Vetting Mentors

Depending on the nature of your program, screening will be different. We recommend interviewing all potential youth mentors - either in person through group interviews or via a video chat tool like Skype or Google Hangouts. It's important to know who's going to be spending time with your class. We also ask for volunteers to have a police reference check. Again, the requirements will vary by organization so ask your administrator for more information.

Training Mentors

Training mentors is an important part of the experience. We want to make sure they understand their role and have some familiarity with the content.

There are two key components to our training:

1. Reading our Mentor Handbook

Ask all volunteer mentors to review our Mentor Handbook.
You can download it here: bit.ly/mentor-handbook

2. Review of the Lesson Plan

We don't expect that mentors are experts in all of the tools and languages we use, but familiarity and experience before they mentor is critical. Consider sending any of the mentors for your programs the coding challenges you'll be running in your class in advance of the class so they can try them themselves.

SETTING THE TONE

Before diving into the lesson plan, it's important to set the tone for learners by going over a few key concepts and values to guide their learning. We like to encourage and instill a growth mindset or 'fail-forward' approach in learners. Acknowledging upfront that technology fails and that's part of the learning process is critical in setting this tone. Encourage exploration, trial and error and collaborative problem solving. And among all else? Patience. Learning to code is like learning a new language and that takes time, patience and lots of practice.

Here are some common things we like to address in all of our programs:

Technology Fails

If a computer isn't working the way it should and a learner is feeling frustrated, acknowledge that this is common. Try to troubleshoot the problem together. Ask what they would do if this happened at home. Would they quit and reopen the program? Restart the computer? If the problem does not resolve itself, try using Google as a resource! It is important to show the learner how to be resourceful for the future.

Student-Driven Problem Solving

Encourage your students to ask others for help first, before coming to you. Often other learners can troubleshoot many of the tech challenges and it's a great opportunity to empower them as leaders. You can also consider creating a troubleshooting checklist with your students. And when all else fails? Google is your friend! Resourcefulness and learning to access information on the Internet is an extremely valuable skill for everyone to learn and it's strongly encouraged.

Inquiry-Based Learning

We've developed many challenges to introduce fundamental programming concepts to youth but we stress that these are just starting points. Allowing students to run with their ideas and questions is an important part of the learning experience.

DIVERSITY MATTERS

It's a fact that the **technology industry has a diversity problem**. We've developed our youth programs and content with diversity in mind. Engaging and supporting a diverse group of kids in technology is critical if we want to close the diversity gap.

Tips for engaging girls:

Connect coding and technology to meaningful and creative projects

Choose and/or adjust challenges that resonate with your girls' interests - do they like music? Art? Animals? Philanthropy?

Connect coding and technology to meaningful and creative career paths

Technology jobs aren't just one job - there are thousands of jobs. Developers work on movies, games, medical devices and more. Sharing these diverse and creative roles can help broaden their understanding of the industry.

Maintain a social and collaborative learning environment

Encourage group work, peer-to-peer mentoring and recognition and demonstration of work and accomplishments. Facilitate ice-breaker activities and games to develop bonding among the participants.

Maintain a strong female mentorship presence

Consider recruiting volunteers or inviting women in the industry as guest speakers and mentors.

Focus on why, not just how

Looping back lessons to why coding matters - not just how to code. Girls really resonate with how what they are working on can have an impact and change the world.

Be aware of unconscious bias

Keeping biases in check is important. We often unintentionally guide boys towards 'boy' things and girls toward 'girl' things. There are subtle biases in society that affect students like 'girls aren't good at math'.

Be aware of imposter syndrome

Imposter syndrome is strong for women and girls and often associated with high achieving students. These are feelings of not being smart, successful or a good student and instead that you are only imposing as such. We have to work extra hard to reinforce girls' skills and aptitude especially for subjects like math, science and technology.

GETTING STARTED

Get excited

Be inspiring

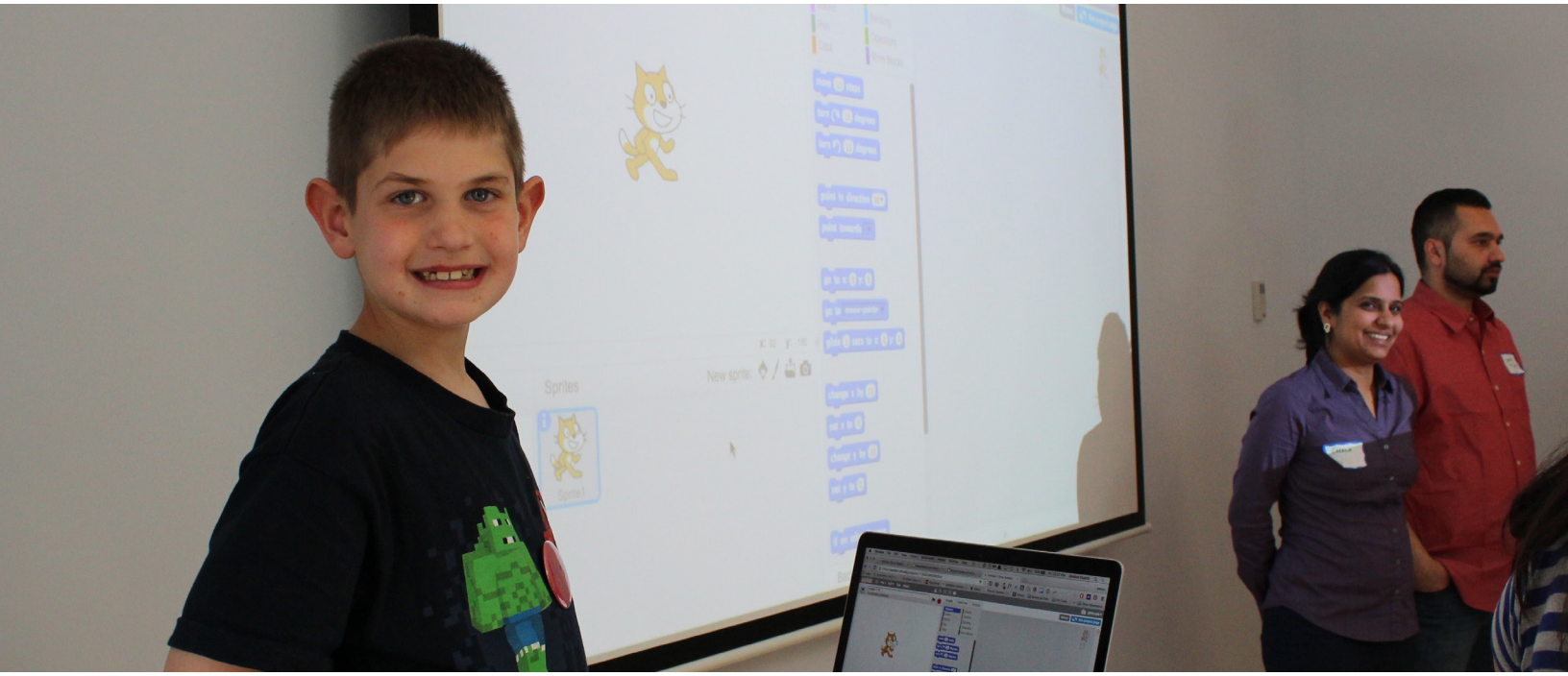
Have an open-mind

Be confident

Be resourceful

Stay positive

Have fun!



KEY PROGRAMMING CONCEPTS

Here are some key programming concepts to help you understand Scratch and programming in general a bit better. You'll notice many of these concepts come up in other subjects and daily life and aren't as foreign or daunting as you might think!

Variable

stores a piece of information i.e. score of a game that increases by 1 value for each goal

Array

allows you to store more than just one piece of information

Function

a type of procedure or routine that performs a distinct operation. There are often 'canned' functions that exist already like the 'jump' block

Sequence

identifying a series of steps for a task. Computers and Scratch read and perform commands in order from top to bottom

Events

one thing causing another thing to happen i.e. 'when clicked' block

Conditionals

making decisions based on conditions i.e. if some condition is met do something, else do nothing or something else

Loops

running the same sequence multiple times i.e. repeat or forever blocks

Boolean Logic

and, or, not are examples of boolean logic. they are values that can be either true or false

Parallelism

making things happen at the same time

Operators

mathematical and logical expressions i.e. X+X block

Debugging

finding problems in code and solving them

Remixing

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

Modularizing

exploring connections between the whole and the parts

Syntax

the spelling or grammar of a programming language. Scratch's blockly structure removes the need for syntax

Algorithm

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

States

'state' in a programming sense is just the same as 'state' in a non-programming sense. i.e. the TV is on or off. variables have states, values don't. for example 42 is 42 and there's nothing you can change about it

RUBY ROBOT

CHALLENGE ONE

LEVEL OF DIFFICULTY ★ ☆ ☆ **TIME** 25 min

THE LESSON

A basic challenge to get students thinking about simple instructions and sequences, or in coding terms, creating algorithms.

1. In groups of two, assign one learner to be "The Programmer" and one to be "The Robot".
2. Assign each pair an activity like tying a shoe or opening a door.
3. Ask the Programmer to explain to their partner (Robot) how to perform the steps needed to complete their activity using words only!
4. Switch pairs
5. After the activity, use this as an opportunity to talk about the importance of simple, clear instructions and sequences.
6. Congratuale learners for creating their first algorithms and pseudo code!

LEARNING OUTCOMES

- Creating basic coding commands
- Algorithms

WHAT TO PREP

☑ Sample activities for the pairs to complete. ie. tying a shoe, opening a door, doing the macarena

EXTENSIONS & MODIFICATIONS

- Have students write down their code and share with other groups to execute
- Choose some code to review as a group - do students recognize any patterns? Any ways they could simplify their algorithms?

all grades | sequences | commands

THE TOOL

Our coding challenges are built around the graphical programming language **Scratch** but there are a lot of others out there like Pencil Code, Touch Develop and Blockly that allow users to write programs. What all these graphical languages have in common is that they allow users to drag and drop blocks of instructions to write programs. Scratch eliminates the need to write any fancy code (or syntax) and get bogged down in nuances of the syntax like colons, commas and paranthases that early on can be a distraction to learning the key concepts. It's because of this that Scratch is a great tool for beginners and youth setting out to learn code.

We chose to use Scratch because we've had a lot of success with it in our programs and there's a really strong community across the world of educators using the tool. We also like it because you can use it in the browser (web-based version) or download the Offline Editor onto computers (use it locally). Not having to rely on internet access is a huge bonus!

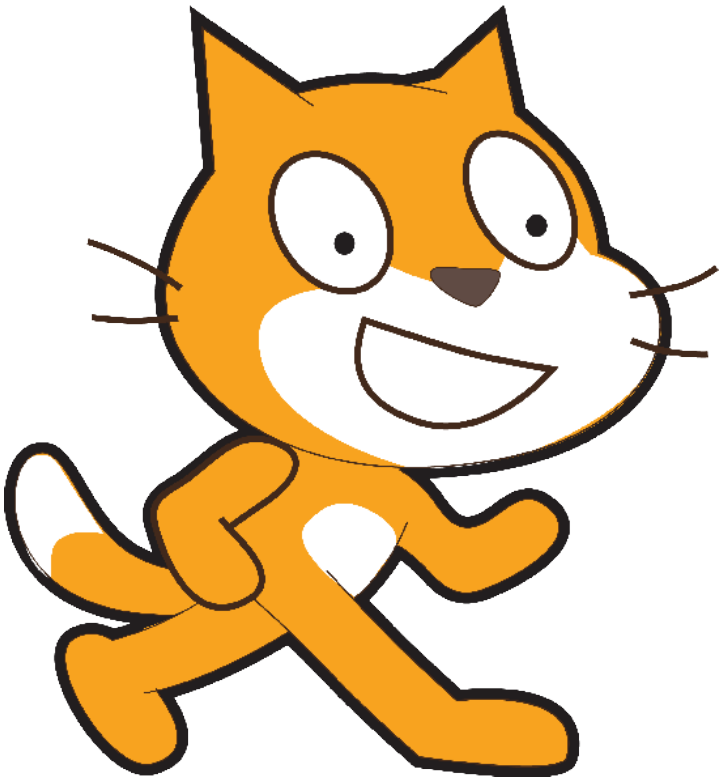
Ultimately, Scratch is designed with learning and education in mind. It's a great collaborative program for drawing art, playing music, and creating games. It is also a place to experiment with mathematical functions, geometry, graphing, webpages, simulations, and algorithms.

LEARN BY DOING

From our experience working with thousands of educators, adult learners and youth, the best way to get started with Scratch and learning to code is by using the tool and trying coding challenges yourself.

Over the following pages we've included basic challenges that will introduce you to the tool and how it work as well as some basic programming concepts. You can use these challenges to familiarize yourself and/or use them as challenges in your classroom to get started with coding.

First, to access Scratch visit: scratch.mit.edu



INTRO TO SCRATCH

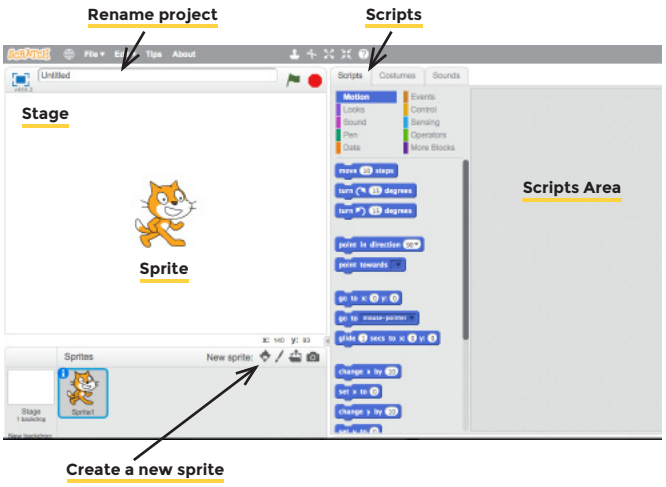
CHALLENGE TWO

LEVEL OF DIFFICULTY ★ ☆ ☆ TIME 10 min

THE LESSON

In this challenge, we will introduce Scratch, how to start a new project and the key features.

1. Visit scratch.mit.edu
2. On the homepage, click on 'Join Scratch' (if you already have an account, click on 'Sign In').
3. Once you're signed in, click on 'Create' to start a new project.
4. Time to explore! Drag and drop blocks (scripts) into the Scripts Area and see what happens.



5. Don't forget to save your work. There's also tips and tricks available from Scratch under 'Tips'.

LEARNING OUTCOMES

- Familiarize yourself with how to access Scratch, create a new project and use the editor

WHAT TO PREP

☑ Scratch online accounts require an email address - plan in advance if each student will be creating their own or if you will create one account for your class that each student will use

☑ Consider printing and posting login details. See 'Printables' section for a template

EXTENSIONS & MODIFICATIONS

- Consider walking through this process with students step-by-step
- If students are quick to catch on, give them 10 minutes to explore and make something happen to the Scratch cat
- Encourage students to work together, ask each other for help, and share what they are discovering

all grades

SCRATCH EDITOR BASICS

Key components of the Scratch editor:

Sprites

Each object in Scratch is called a Sprite. You can add sprites by choosing from the library, painting your own, uploading an image or taking a picture from webcam.

Stage

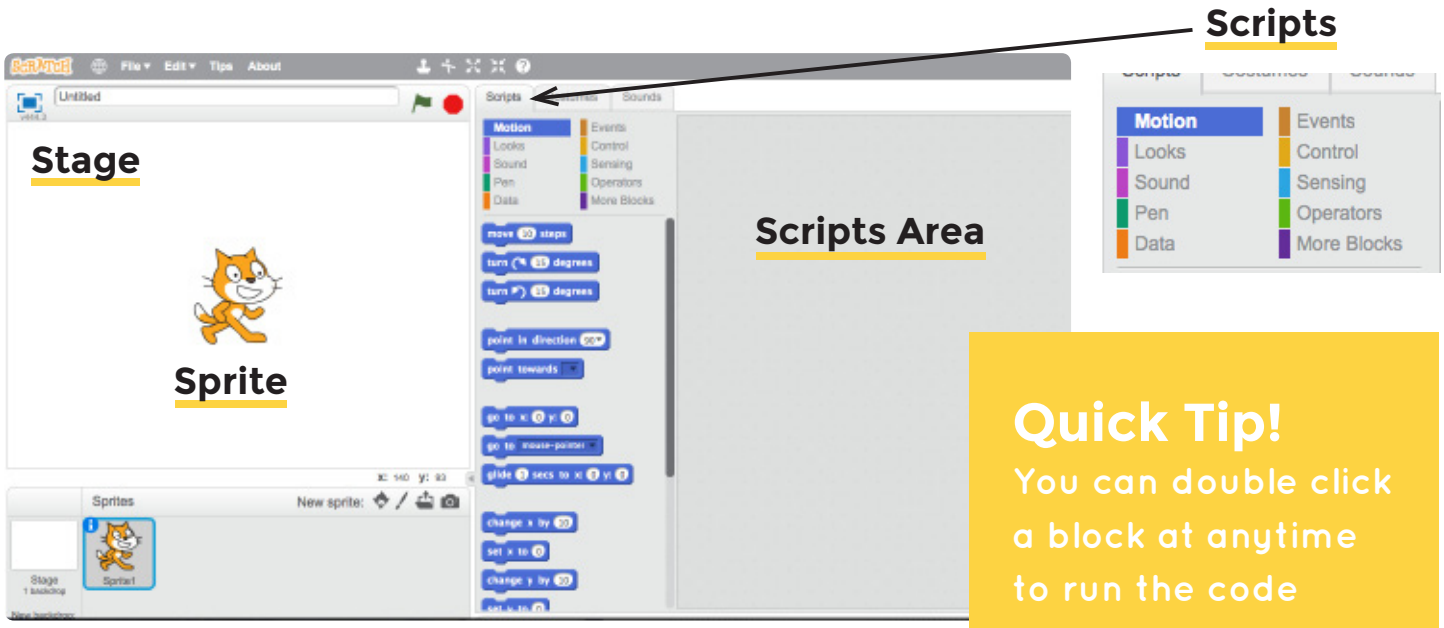
This is where you can preview what your code does. You can add backdrops to your stage. You can 'Start' and 'Stop' the preview at any time using the green flag or red stop symbol at the top of the stage.

Scripts

Scripts are commands. This is how the magic happens. Students will use scripts to instruct their Sprite to do what they want it to do. From moving forward, to saying 'Hello', scripts are arranged in a logical sequence to 'program' the sprite. We'll cover each of these in more detail as you work through challenges.

Scripts Area

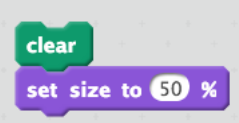
All of the coding happens here. You can drag and drop Scripts into this area. By dragging scripts out of the area you can delete them.



Quick Tip!
You can double click a block at anytime to run the code

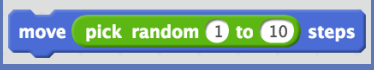
SCRATCH EDITOR BASICS

It is critical that students notice the "hat" (i.e., the round surface) on the top of this block. Other blocks can be connected on both sides but this block is required to be on top. This type of block allows a set of actions to be taken at the dawn of some event.



Blocks are the tools of Scratch programming. You can drag them from their home in toolbox into your script area to interact with them and connect them to each other. You can also change the value in many blocks by clicking the number and typing a new one. Blocks will only snap together when it makes logical or programatic sense to do so.

This block is seemingly meaningless on its own but other blocks can be placed inside of it. The small arrow on the bottom is a hint to its function, alluding to the fact that when the blocks contained within have been executed one by one and the bottom is reached, the program flow returns to the top.



Blocks in blocks in blocks. For example, you can place Operator blocks inside of Motion blocks to create more sophisticated commands like making your sprite move a random number of steps each time.

Making decisions with if. As students complete more complicated challenges and build more sophisticated projects conditional blocks like if<>then will be really important. This block tells the sprite to do something if a certain condition is met otherwise, do something else.

Note: Each script block is explained in more detail in the Scratch Editor itself under 'Tips' 👉 'Blocks' with examples of how to use it.

CODING CHALLENGES

CHALLENGE THREE

LEVEL OF DIFFICULTY ★ ★ ★ **TIME** 20-40 min

THE LESSON

This challenge will introduce students to the basic commands available in Scratch

1. Start by having all students login to Scratch on their computers
2. Have students navigate to the 'Tips' window in Scratch. Instruct students to follow the step-by-step tutorial Getting Started with Scratch to create a dancing cat in Scratch
3. Once students have completed the tutorial, encourage them to experiment by adding other Scratch blocks to make the project their own.
4. Once finished, invite students to share their projects with one another either in groups or in a presentation style.

LEARNING OUTCOMES

- Creating basic coding commands
- Familiarity with Scratch blocks

WHAT TO PREP

☑ Laptops

☑ Login details for Scratch

EXTENSIONS & MODIFICATIONS

- Invite students to record their own sounds
- Invite students to add additional sprites to their project

all grades | sequences | commands

CODING CHALLENGES

CHALLENGE FOUR

LEVEL OF DIFFICULTY ★ ☆ ☆ **TIME** 20 min

THE LESSON: SQUARE DANCE

A basic challenge to get your Sprite moving in a specific pattern.

1. Login in to Scratch and 'Create New'
2. Start by dragging a motion script 'move 10 steps' into the scripts area
3. Press the green start flag. 
4. Nothing happen? (Re)introduce the concept of Events to students. We need something to tell our cat to start moving like a event block.
5. Drag the 'when clicked' blog into the scripts area and run your code
6. What if we want our cat to turn? Try turn block.
7. What if we want our cat to move backwards? Try negative steps.
8. Notice that our cat only moves once per command? We can make our cat repeat their movements using different controls or loops. Try having the commands repeat 10 times or forever.
9. Now, introduce patterns of movement. Instruct students to get their cat to move in a perfect square. You can also use this as an opportunity to introduce the properties of a sqaure = 4 equal sides, 4 90-degree (right) angles...

LEARNING OUTCOMES

- Creating basic coding commands
- Sequences
- Repeat loops

WHAT TO PREP

☑ Laptops

☑ Login details for Scratch

ANSWER KEY

EXTENSIONS & MODIFICATIONS

- What if instead of moving forward the turtle always moved backward?
- What if the cat made only right turns?
- What if we wanted the cat to move in a circle? Or a right angle triangle?

all grades | math | commands | events | loops

CODING CHALLENGES

CHALLENGE FIVE

LEVEL OF DIFFICULTY ★ ☆ ☆ **TIME** 20 min

THE LESSON: DEBUG IT

In this challenge, you'll present students with code that doesn't work and they'll have to troubleshoot what's wrong.

1. Start with a conversation about 'debugging' and how students might tackle finding what's wrong and possible solutions.
2. Present students with the desired outcome and the buggy code. Allow them to troubleshoot to figure out what went wrong. This challenge can be completed on their computers or paper-based and individually, in pairs or small groups.

LEARNING OUTCOMES

- Debugging
- Problem solving

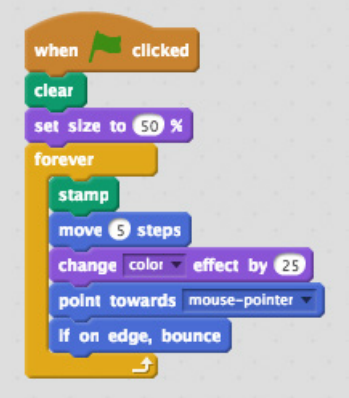
WHAT TO PREP

☑ Laptops

☑ Buggy Code Example - found online here: bit.ly/llc-debug-it

ANSWER KEY

- Missing 'when clicked' + 'if on edge, bounce'



EXTENSIONS & MODIFICATIONS

- Encourage students to make their own code and 'break it' then trade with other students.

all grades | debugging

DEBUGGING SCRATCH

Learning to debug code is an important part of a learners (and educators) coding journey. Here are some common issues we come across in Scratch to help you get started.

How do I delete my blocks?

You can drag the code blocks away off your scripts area. If you want to delete a Sprite or backdrop you can right click on the object you want to delete.

I lost all of my code!

Algorithms are a set of instructions for the computer to do something. Algorithms are common in our everyday lives - a lesson plan is an algorithm for a class or a recipe for making our favourite dish. Writing out step-by-step instructions in plain English is what we call 'pseudo code'.

My code won't work

Code is complex and problems are complex. An important part of computational thinking and coding is breaking down problems into smaller more manageable steps like we might break down a book report into different sections.

My sprite walked off the screen and now I can't get it back, help?

After we break problems into smaller parts, abstraction helps us decide what's important and what's not. It helps manage complexity like a math problem where we decide what information we need to help solve an equation.

The best way to prevent errors and silly mistakes is to have neat, organized code and test your code many times throughout. Spotting errors early can save you lots of time and trouble later!

CONTINGENCY PLANNING

The reality is that **technology fails** - the internet goes down, computers crash, work is lost - and often this feels like failure—not a feeling any teacher likes. And to be honest, neither do students but there are ways to mitigate and be prepared!

Plan Ahead

The more you can plan ahead for potential tech hiccups - the more confident you will be when class starts.

- ☑ Can you set up computers ahead of time?
- ☑ Write logins and passwords on the board. See 'Printables' section for template.
- ☑ Use one universal login versus one per student.
- ☑ Charge laptops before class
- ☑ Can students partner and work together?

Problem Solving & Debugging

We've developed many challenges to introduce fundamental programming concepts to youth but we stress that these are just starting points. Allowing students to run with their ideas and questions is an important part of the learning experience.

Know the basics. There are a few problems that generally account for most of the downtime we have in class. The top two: If the computer won't start, check to see if it's plugged in. If power isn't the problem, restart the computer.

Google the problem. Google is your friend! If there's something wrong with wifi, a keyboard, or Scratch itself - put the issue into Google and see if there's a solution. More often than not there will be. You can also post in the Slack community available when you sign up as an eductor at teacherslearningcode.com

Ask your students. Students are very tech savvy and can often help to troubleshoot one another's problems - it can also turn into a great problem solving activity for the entire class!

Paper-Based Activities

You don't need to use a computer to teach coding! There are a lot of ways to teach fundamental programming concepts like our Ruby Robot Challenge.

NEXT STEPS

Congratulations on completing your first set of challenges! Take a minute to reflect on your experience and note how this might influence how you introduce coding to your students.

- What problems or obstacles did you encounter? How did you solve these problems?
- What obstacles do you anticipate that your students might experience?
- How can you prepare ahead of time to better ensure student success with these introductory challenges?
- What questions could you ask to help students connect what they're learning to the real world?
- What resources will you have available to help students who get "stuck" on a challenge?

Evaluation

In our programs, we take the focus away from evaluation and prioritize creativity, building and the process. With that said, there are a few ways you can evaluate a learner as they work through these challenges.

Programming Concepts

- Did the student show an understanding of programming concepts and use appropriate blocks in their solution?
- Is the student's project or completed code organized, logical and debugged?

Process

- Did the student plan their work?
- Did the student use their time wisely?
- Did the student finish the challenges or complete additional challenges?

Subject Matter & Content

- Did the student make connections between subject concepts and their project?

LESSON PLANNING

Ready to start integrating coding into your classroom?

We've built fun and engaging challenges searchable by subject and grade level that you can choose from to integrate into your classrooms.

For each challenge, we provide a lesson plan to make it easy for you to use in any classroom. Lessons are easily printable or you can save a PDF version onto your USB for quick access later or to share with others.

Each lesson plan includes:

- Challenge Title
- Brief Description
- Level of Difficulty
- Duration of Lesson
- Desired Learning Outcome & Curriculum Connections
- Key Coding Concepts
- How to Prep
- The Lesson | Challenge How-To
- Modifications and Extensions
- Suggested Answer Key
- Other Support Materials like videos and powerpoint presentations

Find lesson plans at
teacherslearningcode.com

OTHER RESOURCES

Scratch Offline Editor
scratch.mit.edu/scratch2download

ScratchEd
scratched.gse.harvard.edu

ScratchJr
scratchjr.org

Debugging Scratch
wiki.scratch.mit.edu/wiki/Debugging_Scripts

Scratch Wiki
wiki.scratch.mit.edu

Scratch Card
scratch.mit.edu/help/cards

Scratch FAQ
scratch.mit.edu/help/faq

Teachers Learning Code Slack Forum
teacherslearningcode.slack.com

Teachers Learning Code website
teacherslearningcode.com

Harvard's CS50 Intro to Computational Thinking + Scratch
Video One + Video Two

PRINTABLES

Over the next several pages you'll find resources created specifically for you to print that will help you promote and run your codeclub. Feel free to print out any or all of this guide as a resource, too!

Printables include:

1. Promotional Poster for Kids & Parents
2. Promotional Poster to Recruit Volunteers
3. Scratch Login Details Handout
4. Certificate of Completion for Learners

KNOW ANY CREATIVE KIDS WHO WANT TO CHANGE THE WORLD?

Start a coding club or introduce coding in your classroom!

※kids learning code ※girls learning code ※teachers learning code

OUR LESSONS AND CODING CHALLENGES WILL HELP KIDS LEARN HOW TO BUILD:

- Video Games
- Animations
- Interactive Art
- Stories
- Music
- . . . and more!

Through hands-on coding challenges that will expose them to new ways of thinking, problem solving and creating.

DATE:

TIME:

VENUE:

FOR MORE INFORMATION & REGISTRATION, CONTACT:

INSPIRE THE NEXT GENERATION OF TECHNOLOGISTS

Become a mentor at one of our code clubs!

※kids learning code

※girls learning code

※teachers learning code

WHAT'S A MENTOR?

- You are either technical or not but have a passion for technology and learning
- You want to learn alongside the kids and help them with their projects
- You encourage an open, flexible learning environment
- You maintain an optimistic view of technology
- You empower kids to become confident, inspired creators

BENEFITS OF MENTORING

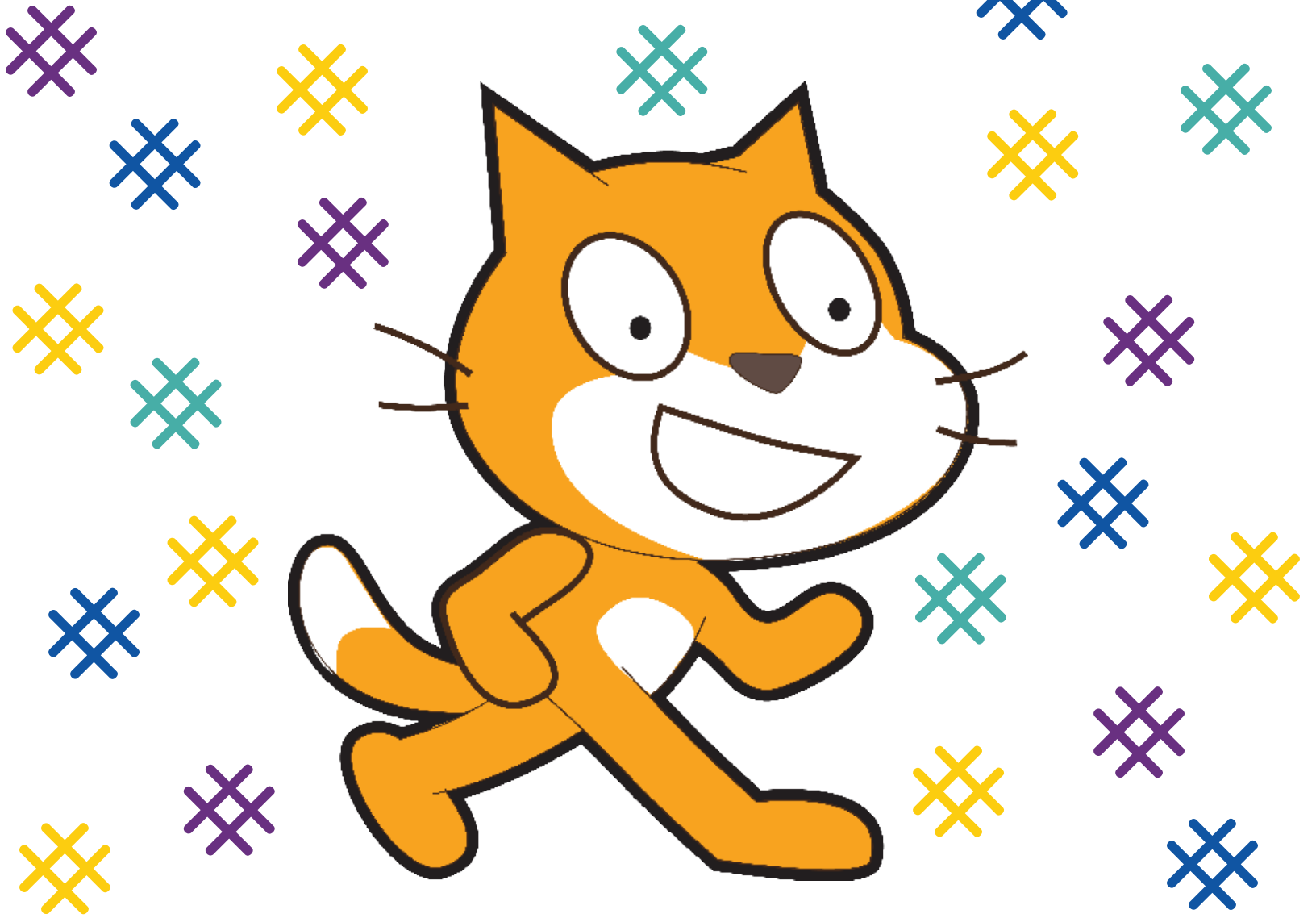
- Experience that "warm and fuzzy" feeling!
- Feel inspired
- Network with other mentors and/or grow your team
- Learn a new tool / language
- Develop leadership and communication skills
- Gain teaching experience
- Give back to the community

DATE:

TIME:

VENUE:

FOR MORE INFORMATION & REGISTRATION, CONTACT:



SCRATCH.MIT.EDU

CLICK ‘SIGN IN’

USERNAME:

PASSWORD:

CONGRATULATIONS

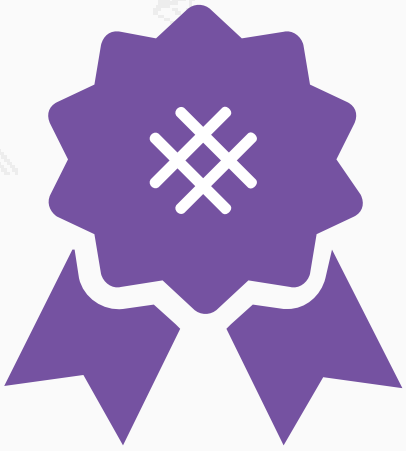
This certificate is awarded to

_____

for the successful completion of our code club

girls learning code

kids learning code



Thank you for helping us inspire the next generation of technologists!

GET IN TOUCH:

Ladies Learning Code
483 Queen Street West, 3rd floor
Toronto, ON • M5V2A9
info@ladieslearningcode.com
teacherslearningcode.com