Learning for the Digital World:

A Pan-Canadian K-12 Computer Science Education Framework

A NOTE FROM THE ADVISORY GROUP AND THE FRAMEWORK ENGAGEMENT AND DEVELOPMENT TEAM

We couldn't be more excited about launching *Learning for the Digital World: A Pan-Canadian K-12 Computer Science Education Framework*. Although 9 in 10 Canadians think that it is important to learn Computer Science, many students do not have the opportunity to do so. This Framework sets out to change that.

As digital technologies play ever-more important roles in our lives, it is critical that all students, especially those who have been traditionally underrepresented in tech—namely women, visible minorities, Indigenous people, and people living in rural and remote areas—have the opportunity to learn foundational skills and competencies to meet the needs of their time. It is essential that we empower all students to harness the power of these new tools. In addition to providing students with the ability to create their own digital projects, Computer Science education also equips students with the skills and competencies they'll need to be more thoughtful and critically minded consumers of digital technologies.

About 18 months ago, we convened a group of policymakers, educators, curriculum developers, and Computer Science experts and began to sketch out a roadmap for developing this Framework. Since then, we have interviewed dozens of people, published two Working Documents, hosted 10 workshops across the country, and launched two national surveys. In total, we've heard from over 650 people living in all corners of Canada.

The journey to this Framework has also energized and inspired us, and solidified our commitment to work towards a future in which all students learn Computer Science. By creating greater alignment in what students learn and promoting more equitable access to high-quality Computer Science education, this Framework is an important first step. Moving forward we're going to work to turn these ideas into action, by inspiring more people to introduce Computer Science education into classrooms across the country and by providing resources and training to educators.

On behalf of the Advisory Group and the Framework Engagement and Development Team, thank you to everyone who helped shape the Framework. We hope you'll join us as we continue our journey.

[illegible]

Melissa Sariffodeen
Chair, Framework Advisory Group and Engagement and Development Team
CEO, Canada Learning Code

TABLE OF CONTENTS

Computer Science Education for Everyone

DIGITAL SKILLS FIT FOR OUR TIME

Public education exists to provide all students with the opportunity to learn the skills and competencies they'll need to meet the needs of our times.

Today, our times are shaped by pervasive digital technologies that are changing how people live, work, interact, and learn. And though the best digital technologies are almost effortless to use, few of us understand how they work, or the logic that underpins them.

As digital technologies are increasingly embedded in our lives, future generations will need to be equipped with the technological skills and competencies to solve present-day and future problems.

We want to empower all students to harness the power of these new tools as both creators and consumers of digital technologies. In addition to providing students with the opportunity to create their own digital projects, Computer Science education also equips them with the skills and competencies they'll need to be more thoughtful and critically minded consumers of digital technologies.

And ultimately, while we prepare our students for the digital world, we can inspire everyone to embrace these new skills and competencies and foster a love of life-long learning.

INEQUITABLE ACCESS TO COMPUTER SCIENCE EDUCATION

Over the past decade, education leaders across Canada have been making strides towards including Computer Science education in curricula. Yet, the landscape remains uneven.

Some provinces and territories offer students the opportunity to learn Computer Science in elementary and middle school, whereas others offer these classes as electives in high school. For some students, Computer Science education is integrated across a variety of subjects, while for others it is an extracurricular activity. And in a few cases, there is little or no opportunity for students to learn the subject.

Moreover, we lack agreement on what foundational Computer Science skills and competencies all students should be learning and when students should learn them. In some instances, students are taught how to create simple programs. In other instances, students are encouraged to learn about about data and digital citizenship. And at times, the only courses offered are for students interested in pursuing more advanced degrees or jobs in the computing sector.

The result is inequitable access to high-quality and comprehensive foundational Computer Science education for all students across Canada.

MORE OPPORTUNITIES TO LEARN COMPUTER SCIENCE ARE NEEDED

2/3
of Canadians support including more Computer Science in school curricula.

70%
of Canadians agree that learning Computer Science is relevant for today and for the future.

90%
of survey respondents agreed that all students should learn Computer Science in elementary and middle school.

7 of 13
provinces and territories include Computer Science skills and competencies in their elementary or middle school curricula.

PRESENTING A VISION FOR COMPUTER SCIENCE EDUCATION IN CANADA

One way to make access to Computer Science education more equitable is to present a vision for our future which would support greater alignment in Computer Science education outcomes for provinces and territories across Canada.

Intended to support the work of policymakers, curriculum developers, and educators interested in implementing Computer Science education in classrooms across the country, the Framework will:

Present a comprehensive vision for Computer Science education in Canada.

Propose a series of foundational skills and competencies that all students would benefit from learning.

Describe possible learning pathways to support student learning across different proficiency levels.

OTHER ENABLERS OF SUCCESS

To fully realize the Framework's potential and ensure all students have equitable access to high-quality Computer Science education, schools and educators will also need to have access to:

1. **Computers, devices, and software:** Although many of the concepts, skills, and competencies included in this Framework can be taught without digital tools, applying the ideas and building digital projects ultimately brings Computer Science to life and allows students to develop the foundational skills and competencies they'll need to meet the needs of our time.
2. **Internet connectivity:** All schools should have reliable Internet access. Unreliable Internet connectivity still affects many parts of Canada, disproportionately excluding northern, rural, and Indigenous students from opportunities to learn Computer Science. Although students can learn many of the skills and competencies without connecting to the Internet, connectivity opens up opportunities for students to collaborate digitally and apply their learnings.
3. **Accessible tools and resources:** Schools should have access to assistive and adaptive electronic devices, accessible software, and universally designed teaching resources to enable all students, including those with physical, cognitive, or learning disabilities, to participate in Computer Science learning opportunities.
4. **Training opportunities and support:** All educators, regardless of the subject areas they teach, should be provided with the time and resources needed to participate in training opportunities and receive ongoing support to learn foundational Computer Science skills and competencies and explore how to integrate Computer Science into their classroom.

Building the Framework

VALUES GUIDING THE DEVELOPMENT OF THE FRAMEWORK

Participatory: The Framework should be built by engaging with a broad range of people, including Computer Science experts, educators, curriculum developers, parents and students, policymakers, tech industry leaders, and non-government organizations (NGOs).

Approachable: The Framework should demystify Computer Science education and encourage more educators and students to explore the field. It should provide a vision for what all students would benefit from learning in clear and simple language.

Inclusionary: The Framework should promote a vision which broadens access to Computer Science education. Computer Science should be taught or learned by anyone, anywhere in Canada regardless of location, age, gender, race, ethnicity, ability, or access to technology.

Integrated: Computer Science is its own distinct field, but it also shares many concepts and ways of thinking with a broad array of subject areas, including civics, languages, math, and science. The Framework should highlight these connections and encourage educators to integrate Computer Science into other subjects.

Human-Centred: Computer Science is about humans using computing to solve a range of challenges. The Framework should highlight the ways in which people and technology have mutually shaped each other, and emphasize that Computer Science is best taught, learned, designed, and understood with a variety of human experiences in mind.

Adaptable: The Framework should promote a vision of Computer Science education that can be adapted to provincial, territorial, and local contexts as well as to emerging technological changes. This will be necessary to ensure it is effectively taught and learned in classrooms across the country and is as future-proof as possible.

METHODS FOR DEVELOPING THE FRAMEWORK

Working with an Advisory Group of leading thinkers from across the country, Canada Learning Code spearheaded the development of this Framework by engaging with a wide range of people, including educators, policymakers, parents, students, industry leaders, and NGOs.

Convened an Advisory Group

We convened 12 people with expertise in different areas, including policymaking, education, and Computer Science, to share their insight and provide advice as we developed the Framework and launched our Working Documents. (See Appendix A to learn more about our advisors.)

Conducted Research

We completed a scan of provincial and territorial curricula, researched similar frameworks from other jurisdictions, and spoke with policymakers, educators, Computer Science experts, and individuals working in the technology sector.

Published Two Working Documents

We published two Working Documents that we circulated for public feedback. The First Working Document presented a vision for Computer Science Education in Canada. The Second Working Document built off this work and introduced a competency guide.

Engaged Canadians

We published two national surveys and organized 10 workshops across the country to collect input and receive feedback. In total, we've heard from over 650 Canadians from across the country working in a variety of different sectors.

FOUR ENGAGEMENT PHASES

Over 650 people from across Canada shared their ideas for the future of K-12 Computer Science Education.

1

Fall 2018

Expert Forum

We convened experts in Computer Science, as well as in policy and curriculum development, to discuss the need for a Computer Science Education Framework, what it could include, and how it should be built.

2

Winter/Spring 2019

Informant Interviews

We spoke with key informants from Canada and around the world to gather recommendations on how to develop the Framework and what to include.

3

Summer/Fall 2019

Stakeholder Engagement

We published an online survey, continued to speak with key informants, and hosted workshops across the country to collect feedback on our First Working Document and gather new ideas.

4

Winter/Spring 2020

Stakeholder and Public Engagement

We published a second online survey and organized a series of online workshops to gather feedback on our Second Working Document.

Our Vision for K-12 Computer Science Education

ALL STUDENTS SHOULD BE ABLE TO HARNESS THE POWER OF TECHNOLOGY

As avid users of digital technologies, all students would benefit from learning foundational skills and competencies in Computer Science, regardless of whether they want to pursue advanced studies or a career in Computer Science or a related field.

All students should should be able to:

Create their own digital projects
Students should understand how technology works and be capable of creating their own digital tools.

Critically assess how technology works and shapes our world
Students should be able to critically assess and speak about the digital tools and technology they use and create.

Use technology to improve our world
Students should be able to use technology to develop creative solutions that can address challenges they personally face, or issues affecting their community and the world around them.

THE FIVE MAIN FOCUS AREAS OF COMPUTER SCIENCE EDUCATION

Computer Science education is more than just coding. It's about helping students to become better creators and consumers of digital technologies.

Although learning how to build digital projects is a key part of Computer Science education, students should also learn a wider set of skills and competencies that will help them to harness the power of digital technologies as both creators and consumers.

A comprehensive approach to K-12 Computer Science education includes learning about the following five focus areas:

Programming

Computing and Networks

Data

Technology and Society

Design

CAPACITIES AND DISPOSITIONS

In addition to teaching important skills and competencies, a comprehensive approach to Computer Science education will also equip students with the capacities and dispositions required to meet the needs of our times.

Discovery: Computer Science education inspires students to approach problems with curiosity and a sense of discovery. It will encourage students to try new things, approach tasks with a growth mindset, and iterate as they master new skills. Students should develop a spirit of experimentation and life-long learning.

Critical Thinking: Computer Science education helps students to develop better mental models of what computers can and cannot do. Students will also develop an understanding of how digital technologies have both shaped and been shaped by societies, equipping them to critically engage with the social, legal, ethical, and political implications of technology.

Perseverance: Computer Science education encourages students to become more comfortable with taking risks, making and accepting mistakes, and learning through experience. By stressing the importance of continuous and unexpected learning, students will learn to move forward despite setbacks, see opportunity in failure, and become more resilient.

Creative Problem Solving: Computer Science education encourages students to explore their creativity and think outside the box. It promotes inventive and flexible solution-seeking, a common Inuit education principle that encourages students to develop innovative solutions to address issues that affect them, their community, and the world.

Collaboration: Computer Science education encourages students to work in teams and to collaborate with others outside their team. Through this, students will build strong communication skills and develop empathy for the perspectives of others. Moreover, it encourages students to constructively give and receive feedback and fosters a willingness to seek help from, share with, and learn from others.

Citizenship: Computer Science education helps students understand the ways in which technology can positively impact society, helping them to become technological stewards who will harness the power of technology to improve the world around them.

K-12 Computer Science Education Competency Guide

THE COMPETENCY GUIDE AT A GLANCE

The competency guide presents the skills and competencies that all students would benefit from learning.

Each Focus Area includes several themes, each of which has a corresponding skill and competency and some suggested learning pathways* to begin introducing these skills and competencies as early as kindergarten.

Programming

- Algorithms
- Data Structures
- Modularity
- Modelling & Abstraction
- Debugging

Computing and Networks

- Hardware & Software
- Connected Devices
- Troubleshooting
- Digital Connectivity
- Cybersecurity

Data

- Storing Data
- Collecting, Organizing, & Visualizing Data
- Modelling & Inferencing
- Applications of AI & Machine Learning
- Data Governance

Technology and Society

- Social Impacts of Digital Technologies
- Digital Communication
- Ethics, Safety, & the Law
- Technology & the Environment
- History of Technology
- Technology & Wellbeing

Design

- Program Design
- User Design
- Visual Design
- Universal Design

* See appendix B to learn more about how to read our learning pathways.

HOW TO READ THE COMPETENCY GUIDE

Confused about a term?
Check out our glossary in Appendix C (page 38).

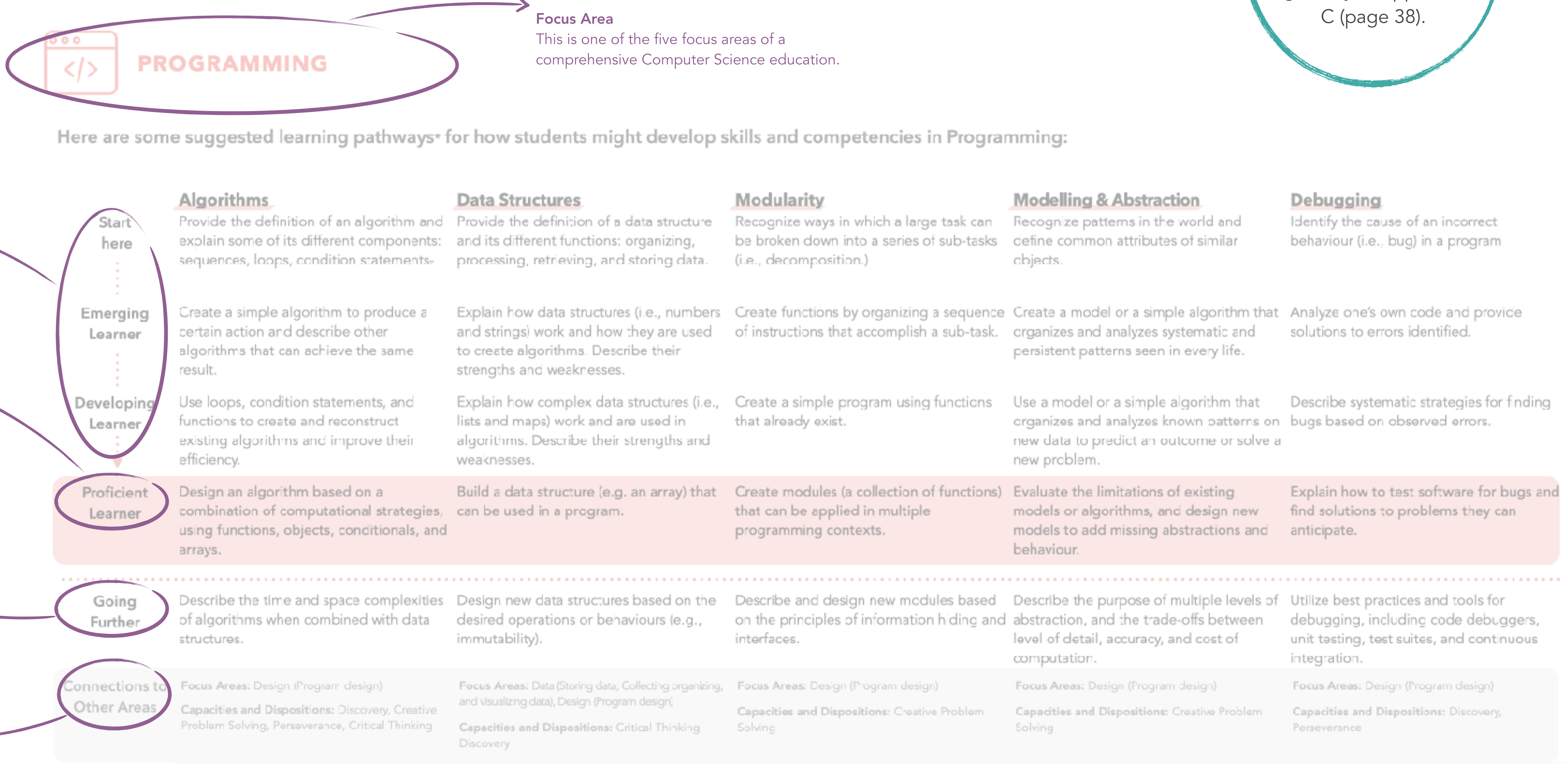
Focus Area
This is one of the five focus areas of a comprehensive Computer Science education.

Suggested Learning Pathways
These pathways provide suggestions for how a student might progress through a particular theme.

Skills and competencies all students should develop
These are the skills and competencies that all students should develop by grade 12.

Going Further
There's always more to learn. These are suggestions for how students can continue their learning.

Connections to other Focus Areas and Capacities and Dispositions
Many of the ideas in each theme support other elements of the competency guide and help students to develop capacities and dispositions. We've highlighted the strongest connections here.



PROGRAMMING

Here are some suggested learning pathways* for how students might develop skills and competencies in Programming:

	Algorithms	Data Structures	Modularity	Modelling & Abstraction	Debugging
Start here	Provide the definition of an algorithm and explain some of its different components: sequences, loops, condition statements-	Provide the definition of a data structure and its different functions: organizing, processing, retrieving, and storing data.	Recognize ways in which a large task can be broken down into a series of sub-tasks (i.e., decomposition.)	Recognize patterns in the world and define common attributes of similar objects.	Identify the cause of an incorrect behaviour (i.e., bug) in a program
Emerging Learner	Create a simple algorithm to produce a certain action and describe other algorithms that can achieve the same result.	Explain how data structures (i.e., numbers and strings) work and how they are used to create algorithms. Describe their strengths and weaknesses.	Create functions by organizing a sequence of instructions that accomplish a sub-task.	Create a model or a simple algorithm that organizes and analyzes systematic and persistent patterns seen in every life.	Analyze one's own code and provide solutions to errors identified.
Developing Learner	Use loops, condition statements, and functions to create and reconstruct existing algorithms and improve their efficiency.	Explain how complex data structures (i.e., lists and maps) work and are used in algorithms. Describe their strengths and weaknesses.	Create a simple program using functions that already exist.	Use a model or a simple algorithm that organizes and analyzes known patterns on new data to predict an outcome or solve a new problem.	Describe systematic strategies for finding bugs based on observed errors.
Proficient Learner	Design an algorithm based on a combination of computational strategies, using functions, objects, conditionals, and arrays.	Build a data structure (e.g. an array) that can be used in a program.	Create modules (a collection of functions) that can be applied in multiple programming contexts.	Evaluate the limitations of existing models or algorithms, and design new models to add missing abstractions and behaviour.	Explain how to test software for bugs and find solutions to problems they can anticipate.
Going Further	Describe the time and space complexities of algorithms when combined with data structures.	Design new data structures based on the desired operations or behaviours (e.g., immutability).	Describe and design new modules based on the principles of information hiding and interfaces.	Describe the purpose of multiple levels of abstraction, and the trade-offs between level of detail, accuracy, and cost of computation.	Utilize best practices and tools for debugging, including code debuggers, unit testing, test suites, and continuous integration.
Connections to Other Areas	**Focus Areas:** Design (Program design) **Capacities and Dispositions:** Discovery, Creative Problem Solving, Perseverance, Critical Thinking	**Focus Areas:** Data (Storing data, Collecting organizing, and visualizing data), Design (Program design) **Capacities and Dispositions:** Critical Thinking Discovery	**Focus Areas:** Design (Program design) **Capacities and Dispositions:** Creative Problem Solving	**Focus Areas:** Design (Program design) **Capacities and Dispositions:** Creative Problem Solving	**Focus Areas:** Design (Program design) **Capacities and Dispositions:** Discovery, Perseverance

PROGRAMMING

By the end of high school, students should be able to create a simple computer program.

All students will benefit from learning programming skills so that they can create their own digital projects. By understanding how to give a computer a set of instructions that allows it to complete a specific task, students will better understand how computers work. This will help students to become more critical consumers of digital technologies and to develop computational thinking which can be applied to various fields.

To do this, students will need to possess skills and competencies in the following themes:

Algorithms

Design an algorithm based on a combination of computational strategies, using functions, objects, conditionals, and arrays.

Data Structures

Build a data structure (e.g., an array) that can be used in a program.

Modularity

Create modules to separate their computer programs into smaller parts, which can be applied in multiple programming contexts.

Modelling & Abstraction

Evaluate the limitations of existing models or algorithms, and design new models to add missing abstractions and behaviour.

Debugging

Explain how to test software for bugs and find solutions to problems they can anticipate.

PROGRAMMING

Here are some suggested learning pathways* for how students might develop skills and competencies in Programming:

	Algorithms	Data Structures	Modularity	Modelling & Abstraction	Debugging
Start here	Provide the definition of an algorithm and explain some of its different components (e.g., sequences, loops, and conditional statements).	Provide the definition of a data structure and its different functions (e.g., organizing, processing, retrieving, and storing data).	Recognize ways in which a large task can be broken down into a series of sub-tasks (i.e., decomposition).	Recognize patterns in the world and define common attributes of similar objects.	Identify the cause of an incorrect behaviour (i.e., bug) in a program.
Emerging Learner	Create a simple algorithm to produce a certain action and describe other algorithms that can achieve the same result.	Explain how data structures (e.g., numbers and strings) work and how they are used to create algorithms. Describe their strengths and weaknesses.	Create functions by organizing a sequence of instructions that accomplish a sub-task.	Create a model or a simple algorithm that organizes and analyzes systematic and persistent patterns seen in everyday life.	Analyze their own code and provide solutions to errors identified.
Developing Learner	Use loops, conditional statements, and functions to create and reconstruct existing algorithms and improve their efficiency.	Explain how complex data structures (e.g., lists and maps) work and are used in algorithms. Describe their strengths and weaknesses.	Create a simple program using functions that already exist.	Use a model or a simple algorithm that organizes and analyzes known patterns on new data to predict an outcome or solve a new problem.	Describe systematic strategies for finding bugs based on observed errors.
Proficient Learner	Design an algorithm based on a combination of computational strategies, using functions, objects, conditionals, and arrays.	Build a data structure (e.g., an array) that can be used in a program.	Create modules to separate their computer programs into smaller parts, which can be applied in multiple programming contexts.	Evaluate the limitations of existing models or algorithms, and design new models to add missing abstractions and behaviour.	Explain how to test software for bugs and find solutions to problems they can anticipate.
Going Further	Describe the time and space complexities of algorithms when combined with data structures.	Design new data structures based on the desired operations or behaviours (e.g., immutability).	Describe and design new modules based on the principles of information hiding and interfaces.	Describe the purpose of multiple levels of abstraction, and the trade-offs between level of detail, accuracy, and cost of computation.	Utilize best practices and tools for debugging, including code debuggers, unit testing, test suites, and continuous integration.
Connections to Other Areas	**Focus Areas:** Design (Program Design) **Capacities and Dispositions:** Discovery, Creative Problem Solving, Perseverance, Critical Thinking	**Focus Areas:** Data (Storing data; Collecting, Organizing, and Visualizing data), Design (Program design) **Capacities and Dispositions:** Critical Thinking, Discovery	**Focus Areas:** Design (Program Design) **Capacities and Dispositions:** Creative Problem Solving	**Focus Areas:** Design (Program Design) **Capacities and Dispositions:** Creative Problem Solving	**Focus Areas:** Design (Program Design) **Capacities and Dispositions:** Discovery, Perseverance

* See Appendix B to learn more about how to read our learning pathways, Appendix C for a glossary, and Appendix D for a list of tools for educators.

COMPUTING AND NETWORKS

By the end of high school, students should understand and be able to use the tools and devices commonly used to build digital projects.

As computers and the Internet become ubiquitous, all students will benefit from learning how these devices work, how they can be connected with other devices and networks, and the benefits and risks associated with our increasingly connected world.

To do this, students will need to possess skills and competencies in the following themes:

Hardware & Software

Program a physical device that collects and exchanges information between its hardware and software components.

Connected Devices

Model how connected devices collect and exchange information with each other without human interaction (commonly known as the Internet of Things (IoT)).

Troubleshooting

Document the steps followed to solve a problem in a way that allows others to solve a similar technical problem.

Digital Connectivity

Explain and model the relationship between routers, switches, servers, topology, and addressing.

Cybersecurity

Assess the role that people play in creating, preventing, and minimizing the impacts of cyberattacks as well as consider how they affect people and society.

COMPUTING AND NETWORKS

Here are some suggested learning pathways* for how students might develop skills and competencies in Computing and Networks:

	Hardware & Software	Connected Devices	Troubleshooting	Digital Connectivity	Cybersecurity
Start here	Explain the composition of a computer system including physical and non-physical components.	Use a computing device to complete an everyday task and compare different types of devices and explain how they might meet or fail to meet certain users' needs or preferences.	Identify and describe common problems associated with computers, external devices, and networks.	Use the Internet to conduct research, as well as a cloud-computing platform to save files and work collaboratively with others.	Define cybersecurity and create safe passwords using effective criteria.
Emerging Learner	Describe and model how hardware and software work together (i.e., sending, receiving, processing, storing information as bits).	Connect a device to another device through a physical or wireless connection or to a cloud-based platform to extend the device's capabilities.	Use common troubleshooting strategies to solve simple problems with computers, external devices, and networks.	Define the Internet, describe how it is designed, and model how information is broken down and sent through devices over networks.	Describe common types of cyberattacks and identify malicious content (e.g., spam, spyware, viruses, phishing, etc.).
Developing Learner	Describe and model how application software, systems software, and hardware interact together.	Create a system by connecting multiple computing devices through physical or wireless connections.	Research and implement solutions to simple problems with computers, external devices, and networks.	Define and model Internet protocols and explain their importance in connecting billions of devices.	Apply common prevention practices (i.e.,: antivirus software and encryption) that prevent or minimize the impact of cyberattacks.
Proficient Learner	Program a physical device that collects and exchanges information between its hardware and software components.	Model how connected devices collect and exchange information with each other without human interaction (commonly known as the Internet of Things (IoT)).	Document the steps followed to solve a problem in a way that allows others to solve a similar technical problem.	Explain and model the relationship between routers, switches, servers, topology, and addressing.	Assess the role that people play in creating, preventing, and minimizing the impacts of cyberattacks as well as consider how they affect people and society.
Going Further	Create an integrated and embedded system that consists of multiple physical devices that collect and exchange information.	Explore the benefits and drawbacks of working with connected devices that exchange information with each other without human interaction.	Research and implement solutions to complex problems with computers, external devices, and networks.	Assess how hierarchy and redundancy enhance the scalability and reliability of the Internet.	Define major cybersecurity risks and recommend security measures that can be taken to prevent them.
Connections to Other Areas	**Focus Areas:** Programming; Data (Storing Data; Collecting, Organizing, and Visualizing data); Technology and Society (History of Technology) **Capacities and Dispositions:** Critical Thinking, Perseverance, Discovery	**Focus Areas**: Design (User design); Technology and Society (Social Impacts of Digital Technologies) **Capacities and Dispositions:** Critical Thinking, Discovery	**Focus Areas:** Programming **Capacities and Dispositions:** Perseverance, Collaboration, Critical Thinking, Creative Problem Solving, Discovery	**Focus Areas:** Technology and Society (Digital Communication) **Capacities and Dispositions:** Discovery, Critical Thinking, Collaboration	**Focus Areas:** Programming **Capacities and Dispositions:** Citizenship, Critical Thinking

* See Appendix B to learn more about how to read our learning pathways, Appendix C for a glossary, and Appendix D for a list of tools for educators.

DATA

By the end of high school, students should be able to explain how we use computers to create, store, organize, and analyze data.

As access to data grows, all students will benefit from understanding how to effectively harness its powers to make sense of the world around them, as well as to better understand how their actions and activities generate data which can be used by others.

To do this, students will need to possess skills and competencies in the following themes:

Storing Data

Assess the benefits and drawbacks of various storage models, including cloud storage, by considering factors such as cost, speed, reliability, accessibility, privacy, and integrity.

Collecting, Organizing, & Visualizing Data

Develop a simple algorithm or program that allows them to organize and represent a dataset to analyze findings, predict future outcomes, or infer trends.

Modelling & Inferencing

Create a computational model based on patterns observed in data and use it to predict future outcomes.

Applications of AI & Machine Learning

Assess how human biases are embedded within technical systems and artificial intelligence.

Data Governance

Understand and be able to advocate for their data rights and the rights of others.

DATA

Here are some suggested learning pathways* for how students might develop skills and competencies in Data:

	Storing Data	Collecting, Organizing & Visualizing Data	Modelling & Inferencing	Applications of AI & Machine Learning	Data Governance
Start here	Save, retrieve, copy, and delete files from a computing device.	Collect data using an appropriate tool and organize it.	Identify patterns in data, charts, and/or graphs.	Identify commonly used digital tools that incorporate AI (artificial intelligence) and machine learning and acknowledge how such tools help people accomplish tasks that only humans could previously do.	Identify ways their digital or physical activity creates digital data and how to adjust privacy settings on commonly used digital tools.
Emerging Learner	Explain how computers store information in bits and bytes and define what information is stored.	Collect, organize, and present data in at least three different formats and use it to support a claim or tell a story.	Create a simple model that organizes patterns observed in data, charts, and/or graphs.	Define AI and give real-world examples of how it has been used to extract information from data.	Discover who owns the digital data they produce and how it is used. Identify whether that data is open or anonymized to remove their personal information.
Developing Learner	Describe how numbers, text, and media are represented in bits/bytes and stored as files.	Analyze how data collection and visualization/storytelling can be shaped by human motive, perspective, and bias.	Make predictions based on patterns found in data, charts, and/or graphs.	Describe basic algorithms of AI systems and how data and machine learning interact.	Assess provincial and national data governance laws and regulations as well as Indigenous data governance agreements.
Proficient Learner	Assess the benefits and drawbacks of various storage models, including cloud storage, by considering factors such as cost, speed, reliability, accessibility, privacy, and integrity.	Develop a simple algorithm or program that allows them to organize and represent a dataset to analyze findings, predict future outcomes, or infer trends.	Create a computational model based on patterns observed in data and use it to predict future outcomes.	Assess how human biases are embedded within technical systems and artificial intelligence.	Understand and be able to advocate for their data rights and the rights of others.
Going Further	Compare networked information storage systems such as the hyperlinks in the World Wide Web, and blockchain.	Describe the attributes that define big data, including volume, velocity, variety, veracity, and value and consider how big data has transformed our everyday lives.	Identify possible risks of using data to make predictions about the world (i.e., bias, over-fitting, under-fitting).	Explain how machines learn. Discuss specific ethical challenges with machine learning and AI.	Make recommendations for how laws and regulations on data governance and privacy can be improved.
Connections to Other Areas	**Focus Areas:** Computing and Networks (Hardware and Software); Programming (Data Structures) **Capacities and Dispositions:** Critical Thinking, Discovery	**Focus Areas:** Programming (Data structures); Design (Visual Design) **Capacities and Dispositions:** Creative Problem Solving, Discovery, Perseverance	**Focus Areas:** Programming (Abstractions) **Capacities and Dispositions:** Discovery, Creative Problem Solving, Citizenship	**Focus Areas:** Technology and Society (Ethics, Safety and the Law); Programming (Algorithms) **Capacities and Dispositions:** Critical Thinking, Citizenship	**Focus Areas:** Technology and Society (Ethics, Safety, and the Law) **Capacities and Dispositions:** Critical Thinking, Citizenship

* See Appendix B to learn more about how to read our learning pathways, Appendix C for a glossary, and Appendix D for a list of tools for educators.

TECHNOLOGY AND SOCIETY

By the end of high school, students should be able to explore the ways in which technology and society have mutually shaped each other.

As both users and creators of digital technologies, all students will benefit from learning how to assess the effects that digital technologies have had on the world around them, allowing them to become responsible digital citizens capable of harnessing the power of technology to solve problems and respond safely and ethically.

To do this, students will need to possess skills and competencies in the following themes:

Social Impacts of Digital Technologies	Digital Communication	Ethics, Safety & the Law	Technology & the Environment	History of Technology	Technology & Wellbeing
Assess the social effects of computing on various groups, including women, visible minorities, people with disabilities, and Indigenous peoples.	Use a digital tool to collaborate with others and build a simple digital project or complete a class project.	Assess the effects of computer crime, hacking, virus distribution, and other illegal or unethical digital activities on society.	Assess how digital technologies can be used to monitor and promote environmental sustainability.	Trace how technology has evolved in different parts of the world and evaluate the variety of associated impacts.	Develop strategies to harness the power of technology to improve their own physical and mental health and wellbeing.

TECHNOLOGY AND SOCIETY

Here are some suggested learning pathways* for how students might develop skills and competencies in Technology and Society:

	Social Impacts of Digital Technologies	**Digital Communication**	**Ethics, Safety & the Law**	**Technology & the Environment**	**History of Technology**	**Technology & Wellbeing**
Start here	Identify how computers and digital technologies have changed how people live, work, and play.	Use a digital tool to interact with others and describe how digital tools have changed how people communicate with each other.	Identify strategies to protect their personal data and identity online.	Identify the natural resources involved in producing commonly used computing devices.	Identify key figures involved in shaping the history of Computer Science, focusing on the role of Canadians and people who have been traditionally underrepresented in the tech sector.	Identify and apply the school's technology policies. Recognize and reproduce aspects of an ergonomic workstation.
Emerging Learner	Explore how a particular technology has both influenced and been shaped by people, groups, and broader cultural practices.	Trace the history and evolution of social media and assess how it has shaped how people communicate and share information.	Define and apply basic copyright principles. Practice ethical use of others' material by using public domain or creative commons material and properly crediting the source.	Identify and apply environmentally safe and responsible methods for disposing of computing devices. Explain why computing devices must be disposed of in a particular way.	Describe examples of some of the earliest computers, including what they looked like, their purpose, and how they were used.	Identify how digital technologies can be used to improve and impair our physical and mental health and wellbeing.
Developing Learner	Understand how computers, digital technologies, and automation have impacted society.	Explore how digital tools can be used to promote inclusive and exclusionary communities. Identify examples of inclusive online messaging as well as examples of cyberbullying and/or hateful speech.	Explain the privacy concerns related to using personal data to profile and target people or to inform automated decision-making.	Explain the environmental impacts of producing computers, software, and other digital technologies (e.g., waste management, climate, natural resources, etc.).	Analyze the evolution of computers, software, and other digital technologies and the impact they have had on our society.	Assess how specific technologies might be used to improve/impair our health and influence our behaviour.
Proficient Learner	**Assess the social effects of computing on various groups, including women, visible minorities, people with disabilities, and Indigenous peoples.**	**Use a digital tool to collaborate with others and build a simple digital project or complete a class project.**	**Assess the effects of computer crime, hacking, virus distribution, and other illegal or unethical digital activities on society.**	**Assess how digital technologies can be used to monitor and promote environmental sustainability.**	**Trace how technology has evolved in different parts of the world and evaluate the variety of associated impacts.**	**Develop strategies to harness the power of technology to improve their own physical and mental health and wellbeing.**
Going Further	Analyze and evaluate how digital technologies have shaped cultures across the world.	Assess how digital tools can be used to spread misinformation and apply strategies for identifying such misinformation.	Analyze and evaluate how policies governing technology and innovation have shaped and will continue to shape the evolution of digital technologies.	Create a digital project to monitor or promote sustainability or develop a solution to an environmental issue caused by digital technologies.	Predict how the field of Computer Science and Computing might evolve and assess how work and future jobs might be affected.	Explain how digital technologies can be used to influence our behaviour, and develop and implement strategies to minimize negative influences.
Connections to Other Areas	**Focus Areas:** Design (Universal Design); Data (Application of AI and Machine Learning) **Capacities and Dispositions:** Critical Thinking, Citizenship	**Focus Areas:** Computing and Networks (Digital connectivity); Programming **Capacities and Dispositions:** Citizenship, Collaboration	**Focus Areas:** Data; Computing and Networks (Cybersecurity); Design (Universal design) **Capacities and Dispositions:** Citizenship, Critical Thinking	**Focus Areas:** Computing and Networks (Hardware and Software) **Capacities and Dispositions:** Citizenship, Creative Problem Solving	**Focus Areas:** Computing and Networks (Hardware and Software) **Capacities and Dispositions:** Citizenship, Critical Thinking	**Focus Areas:** Computing and networks (Cybersecurity) **Capacities and Dispositions:** Citizenship, Critical Thinking, Creative Problem Solving

* See Appendix B to learn more about how to read our learning pathways, Appendix C for a glossary, and Appendix D for a list of tools for educators.

DESIGN

By the end of high school, students should be able to apply design principles to the digital projects they create.

By learning how to design programs, incorporate the perspective of potential users, and create universally accessible products, students will ultimately create better digital projects that work for a wider range of people.

To do this, students will need to possess skills and competencies in the following themes:

Program Design

Document and justify their own or their team's computational processes when creating a program in a way that allows others to follow and understand.

User Design

Collaborate with others to collect feedback on a digital project they or their team made, identify areas for improvement, and implement changes.

Visual Design

Apply the principles of UI (user interface) design to create a digital project that balances aesthetic design with practical application.

Universal Design

Create a user-friendly project that meets provincial and/or other known accessibility standards and accounts for a wide range of human diversity.

DESIGN

Here are some suggested learning pathways* for how students might develop skills and competencies in Design:

	Program Design	User Design	Visual Design	Universal Design
Start here	Identify the rationale or purpose behind programs and digital tools commonly used. Explain how programs are built for a purpose.	Identify the basic principles of design thinking (i.e., empathy, brainstorming, prototyping, testing, iterating).	Identify the basic principles of visual design (i.e., colour, font, contrast, balance, proportion) in digital and physical content.	Define accessibility and identify common accessibility criteria.
Emerging Learner	Apply the iterative steps teams of professional developers commonly take when creating a program to collaboratively build a simple digital project.	Build empathy maps that explore the range of different user perspectives and needs. Brainstorm solutions that will meet the needs of their users.	Use the basic concepts of visual design to create content in any form (e.g., text, audio, video, animation).	Use a web-based tool to assess a digital project's accessibility.
Developing Learner	Describe the choices they or their team made when developing a digital project. What constraints influenced their decision, what needs did they consider, etc.	Prototype a digital project that accounts for different user perspectives and needs.	Evaluate an interactive website or program and identify how it incorporates principles of good UI (user interface) design (i.e., user control, navigability, accessibility, chunking).	Locate their province or territory's accessibility criteria and evaluate a digital project to assess whether it meets those standards.
Proficient Learner	Document and justify their own or their team's computational processes when creating a program in a way that allows others to follow and understand.	Collaborate with others to collect feedback on a digital project they or their team made, identify areas for improvement, and implement changes.	Apply the principles of UI (user interface) design to create a digital project that balances aesthetic design with practical application.	Create a user-friendly project that meets provincial and/or other known accessibility standards and accounts for a wide range of human diversity.
Going Further	Document and justify the design decisions they or their team made in creating a program in a way that allows others to follow and understand.	Conduct user testing on a product (commonly referred to as A/B Testing) to identify what works best and refine a digital project.	Conduct user testing (commonly referred to as A/B Testing) to enhance or improve a digital project and make it easier for people to navigate.	Use and become familiar with adaptive technology and create a digital project that can be used by an adaptive technology.
Connections to Other Areas	**Focus Areas:** Programming **Capacities and Dispositions:** Collaboration, Perseverance, Creative Problem Solving, Critical Thinking	**Focus Areas:** Programming **Capacities and Dispositions:** Collaboration, Perseverance, Creative Problem Solving, Discovery	**Focus Areas:** Programming **Capacities and Dispositions:** Collaboration, Perseverance, Creative Problem Solving, Discovery	**Focus Areas:** Programming; Technology and Society (Social Impacts of Digital Technologies; Ethics, Safety, & the Law) **Capacities and Dispositions:** Citizenship, Creative Problem Solving, Perseverance

* See Appendix B to learn more about how to read our learning pathways, Appendix C for a glossary, and Appendix D for a list of tools for educators.

INTERESTED IN LEARNING MORE ABOUT COMPUTER SCIENCE EDUCATION?

Whether you're new to Computer Science education or looking for opportunities to sharpen your skills, there are a number of ways you can learn more.

Follow Us and Our Friends

Canada Learning Code is proud to count itself among a number of great organizations working to help students and educators learn Computer Science skills and competencies. See Appendix E for a list of other CanCode funded organizations working to help students and educators learn Computer Science, coding, and digital skills.

Register for a Workshop

Canada Learning Code and many other organizations offer a wide range of workshops for people interested in learning more about Computer Science education. Check out canadalearningcode.ca for more information or click on the links in Appendix E to learn more about workshops being offered by other CanCode funded organizations.

Check Out our Lesson Plans

Canada Learning Code has developed over one hundred lesson plans that you can easily implement in your classroom and has also developed a tool to help educators identify connections with other curricular outcomes in their province's or territory's English, math, and science curriculum.

Appendices: Useful Notes and Resources

Appendix A: Advisory Group Members

APPENDIX: ADVISORY GROUP MEMBERS

Advisory Group Members volunteered their time, expert views, and insights on building the Framework. Together, they provided the Framework Engagement and Development team with strategic advice, worked to ensure that we spoke with the right mix of people throughout the engagement process, and provided advice on our Working Documents.

Anna Klimbovskaia
Co-founder & COO
Diversio
@aklimbovskaia

Cathy Adams
Professor, Faculty of Education
University of Alberta

Jace Meyer
Lead for Indigenous Entrepreneurs
Shopify
@JaceActually

Jesse Dougherty
VP
Amazon Web Services (AWS)
@Jessedougherty

Juliet Waters
Chief Knowledge Officer
Kids Code Jeunesse
@JulietWaters

Melissa Sariffodeen
Co-founder & CEO
Canada Learning Code
@melsariffodeen

Michelle Lin
Computer Science Student
McGill University
@XMichelleLinX

Ryan Oliver
Executive Director
Pinnguaq
@pinnguaq

Sandra McKenzie
Principal
McKenzie Consulting

Shirley Tagalik
Retired Educator
Nunavut Department of Education

Vass Bednar
Regulatory entrepreneur + policy architect
Past Chair, Expert Panel on Youth Employment
@VassB

Steven Floyd
Computer Science and Computer Engineering Teacher and PhD Student in Curriculum Studies
@stevenpfloyd

Appendix B: Learning Pathways

APPENDIX: A WORD ABOUT OUR LEARNING PATHWAYS

Along with identifying the skills and competencies that all students would benefit from learning for each of our focus areas, we've also included a suggested learning pathway for each theme we've identified.

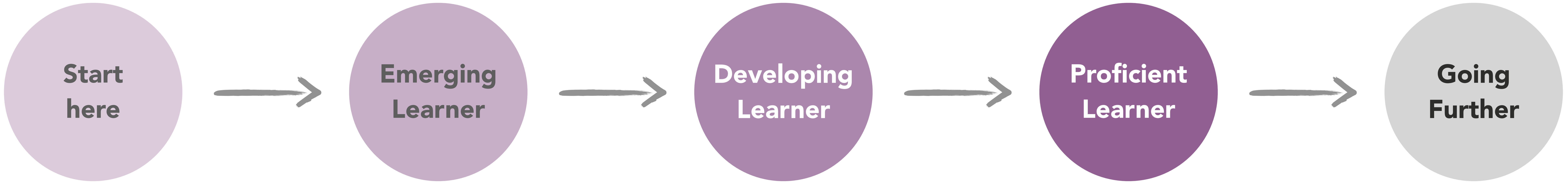
Although we recognize that learning does not always happen in a linear way, these learning pathways provide policymakers and educators with suggestions for how students might progress as they move towards becoming proficient learners.

Learning pathways help to guide the sequence of learning that many students might undertake as their knowledge and understanding of a skill or competency is developed. By ensuring that educators understand what came before and what comes after, learning progressions also help ensure continuity and coherence across grade levels.

Learning progressions are also meant to be living concepts. They should be tested, evaluated, and revised as educators, curriculum developers, and policymakers gain greater insight into how students learn.

APPENDIX: HOW TO READ OUR SUGGESTED LEARNING PATHWAYS

As students work towards becoming proficient learners we've proposed a series of skills and competencies that students would benefit from learning along the way.



Students learn basic skills and competencies and apply new knowledge in simple contexts.

Students build on their knowledge and learn more complex ideas or how to apply their skills and competencies in more complex contexts.

Students continue to build on their knowledge and learn more complex ideas or how to apply their skills and competencies in increasingly more complex contexts.

Students deepen their knowledge and application of the skills and competencies they've been developing.

Learning doesn't end once students have become proficient. So, we've also proposed some ideas for further learning.

A word about grade levels: To ensure that the Framework is adaptable across Canada, the suggested pathways are not tied to any particular grade levels or bands. Students should begin learning the skills and competencies as early as kindergarten.

Appendix C: Glossary

GLOSSARY OF TERMS

A/B Testing:	A method of receiving user feedback by comparing two versions of a single product typically by testing a subject's response to product A against product B, and determining which of the two products is more effective. See also User Testing. [Hanington]
Abstraction:	After problems are broken into smaller parts, abstraction helps decide what's important and what's not. It helps manage complexity, like when we decide what information is needed to help solve a math equation or word problem. [Canada Learning Code]
Accessibility:	The design of products, devices, services, or environments for people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applications (ARIA) standards. [Henry, Abou-Zahra, & Brewer]
Adaptive Technology:	Adaptive technologies are specially-designed tools or technologies that provide enhancements or different ways of interacting with the technology. Adaptive technologies are designed for individuals with a disability or impairment to accomplish a specific task. Unlike assistive technology (see below), adaptive technologies are specifically designed for people with disabilities and would seldom be used by someone without a disability. Wheelchairs are an example of adaptive technology. [Act Center; Center on Technology and Disability]
Algorithms:	A set of instructions that direct a computer to complete some task. Algorithms are common in our everyday lives - a lesson plan is an algorithm for a class, while a recipe is an algorithm for making our favourite dish. Writing out step-by-step instructions in plain English is what we call, 'pseudo code.' [Canada Learning Code]
Array:	Arrays help store related data under one name. An array is a special variable that can store more than one value at a time. All data in an array must be of the same data type. For example, an array called 'dogs' might have the following items: chihuahua, pug, and retriever. [BBC; Canada Learning Code]
Artificial Intelligence (AI):	Sometimes called machine intelligence, AI is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving." [Poole, Mackworth, & Goebel; Russell, & Norvig]
Assistive technology:	Refers to any tool or device that helps people with disabilities perform tasks with greater ease and/or independence. Light signal alerts, and screen readers are just a few examples of assistive technology. [ACT Center]
Automation:	To link disparate systems and software so that they become self-acting or self-regulating. [Ross]
Big Data:	A term used to refer to data sets that are too large or complex to process using traditional techniques. Big Data can be used to produce incredible insights that could not have otherwise been drawn from traditional analytics. These insights help organizations make decisions. [Webopedia]
Bits and Bytes:	At the smallest scale in the computer, information is stored as bits and bytes. Everything in a computer is 0's and 1's. The bit stores just a 0 or 1. One byte typically consists of a collection of 8 bits (e.g., 0 1 0 1 1 0 1 0). [Stanford University]
Cloud-based platform:	Cloud-based platform refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. [Armbrust et al.]
Code:	Any set of instructions expressed in a programming language. [Massachusetts Department of Elementary and Secondary Education (MDESE)]
Computer Bug:	An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. [Tech Terms]

GLOSSARY OF TERMS

Computational Thinking:	Involves identifying a problem and articulating the solution in a way that a computer (or another human) could act on. It involves breaking down big problems into smaller parts, and describing the specific steps needed to overcome these smaller challenges. [Canada Learning Code]
Computing:	Any activity requiring, benefiting from, or creating computers. Computing is a broad field that connects to and draws from many disciplines, including mathematics, electrical engineering, psychology, statistics, fine arts, linguistics, and physical and life sciences processes. [ACM]
Conditionals:	Making decisions based on conditions (e.g., if it's raining, then open your umbrella). [Canada Learning Code]
Cybersecurity:	The protection against access to, or alteration of, computing resources through the use of technology, processes, and training. [TechTarget]
Data:	Information that is collected and used for reference or analysis. Data can be digital or nondigital and can be in many forms, including numbers, text, show of hands, images, sounds, or video. [Computing at School; Tech Terms]
Data Structure:	A particular way to store and organize data within a computer program to suit a specific purpose so that it can be accessed and worked with in appropriate ways. [TechTarget]
Debugging:	Finding problems in code and solving them. [Canada Learning 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. [Canada Learning Code]
Design Thinking:	An iterative approach to solving problems which involves developing a deep understanding of customers' or users' unmet needs. [Naiman]
Digital Project:	Also referred to as a 'digital artifact', a digital project is any type of item produced and stored as an electronic version. Examples of digital projects include digital documents, presentations, programs and codes, video and audio files, images and photographs and the like. [Wikieducator]
Empathy Maps:	A tool used by designers to help clearly understand their end-users and their target audience. It helps create a shared understanding of user needs, and helps in decision making. Traditional empathy maps are split into 4 quadrants (*Says*, *Thinks*, *Does*, and *Feels*), with the user or persona in the middle. Empathy maps provide a glance into who a user is as a whole. [Gibbons]
Encryption:	Encryption is used to protect data in transit, for example data being transferred via networks (e.g. the Internet, e-commerce, mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines). Data encryption translates data into another form, or code, so that only people with access to a secret key (formally called a decryption key) or password can read it. Encrypted data is commonly referred to as ciphertext, while unencrypted data is called plaintext. [Roebuck; Lord]
External Devices:	A computer device, such as a keyboard or printer, that is not part of the essential computer. These auxiliary devices are intended to be connected to the computer and used. Also known as peripheral device. [Webopedia]
Function:	A type of procedure or routine that performs a distinct operation. In this sense, a function is a type of procedure or routine. Most programming languages come with a prewritten set of functions that are kept in a library (like the 'if on edge, bounce' block in Scratch). You can also write your own functions to perform specialized tasks. [Canada Learning Code; Webopedia]
Hardware:	The physical components that make up a computing system, computer, or computing device. [MDESE]
Hierarchy:	An organizational structure in which items are ranked according to levels of importance. [TechTarget]
The Internet:	The global collection of computer networks and their connections, all using shared protocols to communicate. [Computing at School]
The Internet of Things:	A system of interrelated computing devices that transfer data over a network without requiring human-to-human or human-to-computer interaction. [Tech terms]

GLOSSARY OF TERMS

Iterative:	In design thinking, iteration refers to the process of prototyping, testing, analyzing, and refining an idea. [Wikipedia]
Loops:	Running the same sequence multiple times until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things. [Canada Learning Code; Tech Terms]
Machine Learning:	A field of study that gives computers the ability to learn without being explicitly programmed. Machine learning is the process that powers many of the services we use today and helps them make predictions. These include recommendation systems from Netflix, YouTube, and Spotify. Search engines like Google and Baidu, social-media feeds like Facebook and Twitter, and voice assistants like Siri and Alexa also use Machine Learning to make intelligent predictions. [Samuel; Hao]
Model:	A representation of some part of a problem or a system. [MDESE]
Modelling:	The process of creating a model. (see Model)
Modularity:	The characteristic of a software/web application that has been divided (decomposed) into smaller modules. An application might have several procedures that are called from inside its main procedure. Existing procedures could be reused by recombining them in a new application. [Techopedia]
Module:	A software component or part of a program that contains one or more procedures. One or more independently developed modules make up a program. [Techopedia]
Networks:	A group of computing devices (e.g., personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources. [K12CS Framework]
Physical or Wireless Connection:	A physical or wireless attachment between multiple computing systems, computers, or computing devices. [K12CS Framework]
Programming:	The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. [MDESE]
Protocols:	Rules that allow electronic devices to communicate with each other. These rules include what type of data may be transmitted, what commands are used to send and receive data, and how data transfers are confirmed. You can think of a protocol as a spoken language. Each language has its own rules and vocabulary. If two people share the same language, they can communicate effectively. Similarly, if two hardware devices support the same protocol, they can communicate with each other, regardless of the manufacturer or type of device. For example, an Apple iPhone can send an email to an Android device using a standard mail protocol. [Tech Terms]
Prototype:	An early approximation of a final product or information system, often built for demonstration purposes. [TechTarget, Techopedia]
Redundancy:	A system design in which a component is duplicated, so if it fails, there will be a backup. [TechTarget]
Reliability:	An attribute of any system that consistently produces the same results, preferably meeting or exceeding its requirements. [FOLDOC]
Routers:	A device or software that determines the path that data packets travel from source to destination. [TechTarget]
Scalability:	The capability of a network to handle a growing amount of work or its potential to be enlarged to accommodate that growth. [Bondi]
Sequence:	A crucial component of an algorithm. Sequencing refers to identifying a series of steps necessary to complete a task. The sequence can contain any number of actions, but no actions can be skipped in the sequence. In the step-by-step process of an algorithm, the order of these steps is crucial. [Webopedia; Fox, P.]

GLOSSARY OF TERMS

Server:	A computer that provides data to other computers through a network (e.g., the Internet). [Tech terms]
Software:	Programs that run on a computing system, computer, or other computing device. [K12CS Framework].
String:	A sequence of letters, numbers, and/or other symbols. A string might represent, for example, a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring. [TechTarget]
Subtasks:	See decomposition.
Switches:	A high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN). [Techopedia]
System:	A collection of elements or components that work together for a common purpose. [TechTarget]
Topology:	Network Topology refers to the layout of a network and how different nodes in a network are connected to each other and how they communicate. Topologies are either physical (the physical layout of devices on a network) or logical (the way that the signals act on the network media, or the way that the data passes through the network from one device to the next). [Webopedia]
Troubleshooting:	A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computing system. [Techopedia, TechTarget]
Unit Testing:	The type of testing where a developer (usually the one who wrote the code) proves that a code module (the "unit") meets its requirements. [FOLDOC]
User Interface (UI):	A user interface is the part of a computer and its software that people can see, hear, touch, talk to, or otherwise understand or direct. [Galitz]
User Interface (UI) Design:	The design of a user interface in such a way that it is easy for people to understand and use. The goal of user interface design is to create usable and effective interfaces that meet users' needs, capabilities, and limitations. [Galitz]
User Testing:	Usability testing refers to evaluating a product or service by testing it with representative users. The goal is to identify any usability problems, collect qualitative and quantitative data and determine the participant's satisfaction with the product. This is often considered the most critical phase of the design process. Without input from the end-users, designers won't know if their solution is on target or not, and won't know how to evolve their designs to meet their user's needs. See also A/B Testing. [Usability.gov; UserTesting]
Variable	Variables are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so programs can be understood. It is helpful to think of variables as containers that hold information. [LaunchSchool]
Visual Design:	Visual design aims to shape and improve the user experience through considering the effects of illustrations, photography, typography, space, layouts, and colour on the usability of products and on their aesthetic appeal. [Interaction Design Foundation]

REFERENCES

- **Adaptive Computing Technology (ACT) Center. (n.d.).** What is adaptive technology? Retrieved from https://actcenter.missouri.edu/about-the-act-center/what-is-adaptive-technology/
- **Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010).** *A view of cloud computing. Communications of the ACM (Vol. 53, Article No. 4; pages 50-58).* Retrieved from https://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext#:~:text=Defining%20Cloud%20Computing,centers%20that%20provide%20those%20services.
- **Association for Computing Machinery (ACM). (2006).** Computing curricula 2005: The overview report. Retrieved from http://www.acm.org/education/curric_vols/CC2005-March06Final.pdf
- **BBC. (n.d.).** Bitesize: Arrays and lists. Retrieved from https://www.bbc.co.uk/bitesize/guides/zy9thyc/revision/1
- **Bondi, A. (2000).** Characteristics of scalability and their impact on performance. *Proceedings of the second international workshop on Software and performance – WOSP '00 (pages 195 - 203).* Retrieved from https://dl.acm.org/doi/10.1145/350391.350432
- **Canada Learning Code (n.d.).** Teachers Learning Code Digital toolbox: Quick start guide. Retrieved from https://drive.google.com/file/d/1pojSd_3njjMF_nw0tjOAuWW7joZdjBc6/view?ts=5ca798c7
- **Center on Technology and Disability. (n.d.).** Center on Technology and Disability - Assistive Technology Glossary. Retrieved from https://www.ctdinstitute.org/sites/default/files/file_attachments/CTD-ATglossary-v3.pdf
- **Computer Hope. (n.d.).** Retrieved from https://www.computerhope.com/jargon/e/external.htm
- **Computing At School. (2013).** Computing in the national curriculum: A guide for primary teachers. Retrieved from https://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf
- **Fox, P. (n.d.).** The building blocks of algorithms. *Khan Academy.* Retrieved from https://www.khanacademy.org/computing/ap-computer-science-principles/algorithms-101/building-algorithms/a/the-building-blocks-of-algorithms
- **Free on-line dictionary of computing (FOLDOC). (n.d.).** Retrieved from https://foldoc.org
- **Galitz, W. (2007).** The essential guide to user interface design: An introduction to GUI design principles and techniques, third edition. Indianapolis: John Wiley & Sons. ISBN 978-0-470-05342-3
- **Gibons, S. (2018).** Empathy Mapping: The first step in design thinking. *Neilsen Norman Group.* Retrieved from https://www.nngroup.com/articles/empathy-mapping/#:~:text=Definition%3A%20An%20empathy%20map%20is,2)%20aid%20in%20decision%20making.
- **Hanington, J. (2012).** The ABCs of A/B Testing. *Pardot.* Retrieved from https://www.pardot.com/blog/abcs-ab-testing/
- **Hao, K. (2018).** What is Machine Learning? *MIT Technology Review.* Retrieved from https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/
- **Henry, S.L., Abou-Zahra, S., & Brewer, J. (2016).** The role of accessibility in a universal web. *W4A '14 'Proceedings of the 11th Web for All Conference' (Article No. 17; pages 1-4).* Retrieved from https://dl.acm.org/doi/10.1145/2596695.2596719
- **Interaction Design Foundation (n.d.).** What is visual design? Retrieved from https://www.interaction-design.org/literature/topics/visual-design
- **K12 Computer Science Framework (n.d.).** Glossary of terms and definitions. Retrieved from https://k12cs.org/glossary/#jump-to-m
- **Launch School. (n.d.).** What is a variable? Retrieved from https://launchschool.com/books/ruby/read/variables **Lord. (2019).** What is data encryption? Definition, best practices, and more. *Digital Guardian.* Retrieved from https://digitalguardian.com/blog/what-data-encryption
- **Lord. (2019).** What is data encryption? Definition, best practices, and more. Digital Guardian. Retrieved from https://digitalguardian.com/blog/what-data-encryption
- **Massachusetts Department of Elementary and Secondary Education (MDESE). (2016).** 2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework. Retrieved from https://www.doe.mass.edu/frameworks/dlcs.pdf
- **Naiman, L. (2019).** Design thinking as a strategy for innovation. *Creativity at Work.* Retrieved from https://www.creativityatwork.com/arts-in-business-applying-the-arts-to-organisational-learning/
- **Poole, D., Mackworth, A., & Goebel, R. (1998).** Computational Intelligence: A Logical Approach. New York: Oxford University Press. ISBN 978-0-19-510270-3.
- **Porter, K. (n.d.).** What is encryption and how does it protect your data? *NortonLifeLock.* Retrieved from https://us.norton.com/internetsecurity-privacy-what-is-encryption.html
- **Ross, B. (2016).** What is automation and how can it improve customer service? *Information Age.* Retrieved from https://www.information-age.com/industry/software/123461408/what-automation-and-how-can-it-improve-customer-service
- **Russell, S.J., & Norvig, P. (2009).** Artificial Intelligence: A Modern Approach (3rd ed.). Upper Saddle River, New Jersey: Prentice Hall. ISBN 978-0-13-604259-4..
- **Roebuck. (2011).** Encryption: High-impact strategies. Tebbo Publishing. ISBN 978-1-74304-562-6
- **Samuel, A. L. (1959).** Some studies in machine learning using the game of checkers. *IBM Journal of research and development (Vol. 3, Article No. 3; pages 210-229).* Retrieved from https://pdfs.semanticscholar.org/9330/a04e17d3b9ea092bd7dd5295b2d61d53bff5.pdf?_ga=2.188529384.137243211.1594666353-512854151.1594666353
- **Stanford University. (n.d.).** Bits and bytes. Retrieved from https://web.stanford.edu/class/cs101/bits-bytes.html
- **Techopedia technology dictionary. (n.d.).** Retrieved from https://www.techopedia.com/dictionary
- **Tech Terms. (n.d.).** Tech terms computer dictionary. Retrieved from https://www.techterms.com
- **TechTarget network. (n.d.).** Retrieved from https://www.techtarget.com/network
- **Usability.gov. (n.d.).** Usability Testing. Retrieved from https://www.usability.gov/how-to-and-tools/methods/usability-testing.html
- **User Testing. (2018).** IDEO's human centered design process: How to make things people love. Retrieved from https://www.usertesting.com/blog/how-ideo-uses-customer-insights-to-design-innovative-products-users-love
- **UXPin. (n.d.).** The practical guide to empathy maps: 10-minute user personas. Retrieved from https://www.uxpin.com/studio/blog/the-practical-guide-to-empathy-maps-creating-a-10-minute-persona/
- **Webopedia. (n.d.).** Retrieved from https://www.webopedia.com/
- **WikiEducator. (n.d.).** *Defining Digital Artifacts. Retrieved from* https://wikieducator.org/Digital_artefact
- **Wikipedia. (n.d.).** Defining Iterative Design. Retrieved from https://en.wikipedia.org/wiki/Iterative_design

Appendix D: Tools for Educators

DIGITAL AND UNPLUGGED TOOLS FOR EDUCATORS

Raspberry Pi

What is it? A single board computer that allows students to create physical computing projects.

How does it support student learning? It teaches students about physical computing.

Check it out at: www.raspberrypi.org

Voiceflow

What is it? A tool that allows users to collaboratively design, prototype and build Alexa Skills and Google Actions - without coding.

How does it support student learning? It teaches students how to build a digital project using pre-coded blocks.

Check it out at: www.voiceflow.com

Scratch

What is it? A free programming language and online community where you can create your own interactive stories, games, and animations.

How does it support student learning? It helps students learn about Block-based coding.

Check it out at: www.scratch.mit.edu

Little Robot Friends

What is it? An interactive and fun robot that can be programmed and customized by using code.

How does it support student learning? Students learn about coding, electronics, and how they can be programmed.

Check it out at: www.littlerobotfriends.com

Glitch

What is it? A simple tool for creating web applications.

How does it support student learning? Students learn how to make a simple website and learn about HTML (the standard language to create websites).

Check it out at: www.glitch.com

Arduino

What is it? A compact integrated circuit (called a microcontroller) that can be programmed to control lights, motors, sensors, and other physical computing components.

How does it support student learning? Students learn about digital devices and about text-based coding.

Check it out at: www.arduino.cc

DIGITAL AND UNPLUGGED TOOLS FOR EDUCATORS

CoSpaces

What is it? A web and app-based tool that allows students to create interactive 3D virtual and augmented environments and animate them with code.

How does it support student learning? Students learn about block-based coding and about virtual reality.

Check it out at: www.cospaces.io/edu

Trinket

What is it? An all-in-one online coding environment where learners can write and run code from an internet browser.

How does it support student learning? Students learn about educational concepts (like math and physics) by writing code.

Check it out at: www.trinket.io

Pixlr

What is it? A free online photo editor.

How does it support student learning? Students learn about and practice skills in visual design.

Check it out at: www.pixlr.com

Canva

What is it? A free online graphic design platform.

How does it support student learning? Students learn about and practice skills in visual design and learn how to design media graphics, presentations, posters and other visual content.

Check it out at: www.canva.com

Piskel

What is it? A free online editor for creating and animating sprites and pixel art.

How does it support student learning? Students learn how to create and design animations.

Check it out at: www.piskelapp.com

Soundtrap

What is it? A free online music-making tool.

How does it support student learning? Students learn to make music online by using loops and software instrument and by recording vocals, guitars, and more.

Check it out at: www.soundtrap.com

DIGITAL AND UNPLUGGED TOOLS FOR EDUCATORS

TinkerCad

What is it? A free, online, 3D modelling and microcontroller prototyping program.

How does it support student learning? Students can learn design skills, as well as electrical wiring and microcontroller prototyping and programming.

Check it out at: www.tinkercad.com

Lynx

What is it? A free text-based programming language that is available in English, French, Ojibwe and Mi'kmaw.

How does it support student learning? Provides an accessible introduction to text-based programming with the opportunity to connect to a wide variety of provincial curriculum areas.

Check it out at: www.lynxcoding.club

Micro:bit

What is it? A microcontroller which can be programmed to complete a number of tasks.

How does it support student learning? Students can develop programming, electrical engineering and physical computing skills using one of multiple code editors, including MakeCode, Python or Scratch.

Check it out at: www.microbit.org

ScratchJr

What is it? A free introductory programming app that enables young children to create their own interactive stories and games.

How does it support student learning? Students can learn the basic concepts of computer by programming sprites in a block-based, introductory programming language.

Check it out at: www.scratchjr.org

SketchUp

What is it? A 3D modelling computer program for a wide range of drawing applications such as architectural, interior design, and film and video game design.

How does it support student learning? Students can learn about 3D modelling and about basic skills and concepts in design.

Check it out at: www.sketchup.com

BeeBot

What is it? A robot that can be programmed without using a computer.

How does it support student learning? Students can learn how robots are controlled by code, as well as develop an understanding of sequential instructions and spatial reasoning.

Check it out at: www.robotixeducation.ca or www.terrapinlogo.com

DIGITAL AND UNPLUGGED TOOLS FOR EDUCATORS

Repl.it

What is it? A free online programming environment that supports several text-based programming languages including C, C++, Java, JavaScript and Python.

How does it support student learning? Provides students with an opportunity to develop and share projects using a variety of programming languages.

Check it out at: www.repl.it

The Algorithm Literacy Project

What is it? An online project with educational guides, discussion guides, and tools to introduce students and educators to algorithms.

How does it support student learning? Provides students with an opportunity to learn about algorithms, how they influence our digital experiences, and how they impact us.

Check it out at: https://algorithmliteracy.org

Makecode

What is it? A free open source platform for learning how to program.

How does it support student learning? Students can develop programming skills and understand the connection between programmed code and physical devices.

Check it out at: https://www.microsoft.com/en-us/makecode

Art:bit

What is it? A platform that optimizes the micro:bit for LED animation and younger learners. Instantly pairs the micro:bit with Chromebooks and iPads. Fully bilingual.

How does it support student learning? Simplifies exploring with the micro:bit, making it easy for younger learners and elementary school generalists to use.

Check it out at: https://kidscodejeunesse.org/artbit

Appendix E: CanCode Funded Organizations

CANCODE FUNDED ORGANIZATIONS

CanCode aims to equip Canadian youth, including traditionally underrepresented groups, with the skills they need to be prepared for further studies, including advanced digital skills and science, technology, engineering and math (STEM) courses, leading to the jobs of the future. Check out the following organizations that were funded through the CanCode Program to deliver digital and coding skills development initiatives until 2021.

Actua
actua.ca

A.S.T.C. Science World Society (Science World BC)
scienceworld.ca

Black Boys Code (BBC)
blackboyscode.ca

Boys & Girls Club of Canada (BGCC)
bgccan.com

Brilliant Labs
brilliantlabs.ca

CADRE21
cadre21.org

Canada Learning Code
canadalearningcode.ca

COVE
coveocean.com

Cybera
cybera.ca

Elephant Thoughts Education Outreach
elephantthoughts.com

FIRST Robotics Canada
firstroboticscanada.org

Fusion Jeunesse
fusionjeunesse.org

Grandir Sans Frontières
grandirsansfrontieres.org

Hackergal
hackergal.org

Information and Communications Technology Council (ICTC)
ictc-ctic.ca

Kids Code Jeunesse (KCJ)
kidscodejeunesse.org

Let's Talk Science (LTS)
letstalkscience.ca

MediaSmarts
mediasmarts.ca

Pinnguaq Association
pinnguaq.com

Saskatchewan Science Centre (SSC)
sasksciencecentre.com

Saskatoon Industry Education Council
saskatooniec.ca

Science Alberta Foundation (MindFuel)
mindfuel.ca

Science East Science Centre
scienceeast.nb.ca

Science North
sciencenorth.ca

TakingITGlobal
tigweb.org

The Learning Partnership Canada
thelearningpartnership.ca

Ulnooweg, "Digital Mi'kmaq"
ulnooweg.ca/digital-mikmaq

Appendix F: Funders and Project Team

THANK YOU TO OUR FUNDERS AND TEAM MEMBERS

Funders

Canada

amazon

Project Manager

Anna Villanueva

Educator Programs Manager

Canada Learning Code

CANADA LEARNING CODE

Framework Engagement and Development Team

Laurie Drake

Rukhsaar Daya

Rosemary McManus

MASSLBP

We'd also like to thank all the Computer Science, education, and industry experts as well as all Canada Learning Code staff who shared their ideas, reviewed drafts, and provided our team with suggestions and feedback.

“Is that magic?”

“No, we coded it.”