Teacher Training Package

BRITISH COLUMBIA

Introduction:
What's Inside this Package?

This package is designed for educators who will be training Grade 6 to 9 teachers to support student learning in computational thinking and coding. Computational thinking is a content module in the new Applied Design, Skills, and Technologies (ADST) curriculum from Grades 6 to 8. In Grade 9, the ADST curriculum offers text-based coding within the Information and Communications Technologies (ICT) content module.

In particular, this package includes student learning resources for Grade 6 to 9 teachers to support computational thinking and coding in the classroom. The resources are also designed to reinforce the core competencies in the redesigned BC curriculum, as well as the big ideas and curricular competencies specifically in the ADST curriculum.

The student learning resources are organized into 2 units:

1. Introduction to Computational Thinking
2. Continuing Explorations

Each unit includes the following:

- Up to 10 hours of instructional time
- Sequential lesson plans and student activities
- A project-based activity
- Assessments

In support of these resources, this package provides information that you can use to supplement your understanding of the content. First, it gives a brief overview of the core competencies in the redesigned BC curriculum. It then focuses on the ADST curriculum and explores how computational thinking is not only in the ADST curriculum, but also evident in other learning areas across the BC curriculum. Lastly, it introduces computational thinking as a set of strategies for solving problems, and discusses how computational thinking is related to coding.

In addition to the above materials, this package outlines an approach methodology for building your training workshops in your own districts and schools. It also includes guidelines and resources to help you support teachers in adopting computational thinking in their practice.

To summarize, this package consists of the following sections:

BC Curriculum Overview

The redesign of the BC curriculum implemented in 2016 emphasizes personalized, competency-driven, concept-based learning. At the heart of the competencies are the core competencies, which are embedded in all learning areas:

- Creative Thinking
- Critical Thinking
- Communication
- Positive Personal and Cultural Identity
- Personal Awareness and Responsibility
- Social Awareness and Responsibility

Within each learning area, there are three elements that follow the Know-Do-Understand model:

- Content, what students are expected to know
- Curricular Competencies, what students are expected to do
- Big Ideas, what students are expected to understand

Applied Design, Skills, and Technologies Curriculum Overview

The new Applied Design, Skills, and Technologies (ADST) curriculum is an experiential, hands-on program, where students learn through making.

Big Ideas

The Big Ideas capture the practices of applying design processes, skills, and technologies involved in the making process. The Big Ideas for Grades 6 to 9 are shown below.

Big ideas	Grades 6 to 8	Grade 9
Applied Design	Design can be responsive to identified needs.	Social, ethical, and sustainability considerations impact design.
Applied Skills	Complex tasks require the acquisition of additional skills.	Complex tasks require the sequencing of skills.
Applied Technologies	Complex tasks may require multiple tools and technologies.	Complex tasks require different technologies and tools at different stages.

Curricular Competencies

The Curricular Competencies further define the practices outlined in the Big Ideas, and in general, are categorized under the following stages of the making process:

- Understanding Context
- Defining
- Ideating
- Prototyping
- Testing
- Making
- Sharing

The table below lists the Curricular Competencies for Grades 6 to 9 that fall under each stage.

	Grades 6 to 8	Grade 9
Understanding Context	Empathize with potential users to find issues and uncover needs and potential design opportunities.	Engage in a period of research and empathetic observation in order to understand design opportunities.
Defining	Choose a design opportunity. Identify key features or potential users and their requirements. Identify criteria for success and any constraints.	Choose a design opportunity. Identify potential users and relevant contextual factors. Identify criteria for success, intended impact, and any constraints.
Ideating	Generate potential ideas and add to others' ideas. Screen ideas against criteria and constraints. Evaluate personal, social, and environmental impacts and ethical considerations. Choose an idea to pursue.	Take creative risks in generating ideas and add to others' ideas in ways that enhance them. Screen ideas against criteria and constraints. Critically analyze and prioritize competing factors, including social, ethical, and sustainability considerations, to meet community needs for preferred futures. Choose an idea to pursue, keeping other potentially viable ideas open.

Prototyping	Identify and use sources of information. Develop a plan that identifies key stages and resources. Explore and test a variety of materials for effective use. Construct a first version of the product or a prototype, as appropriate, making changes to tools, materials, and procedures as needed. Record iterations of prototyping.	Identify and use sources of inspiration and information. Choose a form for prototyping and develop a plan that includes key stages and resources. Evaluate a variety of materials for effective use and potential for reuse, recycling, and biodegradability. Prototype, making changes to tools, materials, and procedures as needed. Record iterations of prototyping.
Testing	Test the first version of the product or the prototype. Gather peer and/or user and/or expert feedback and inspiration. Make changes, troubleshoot, and test again.	Identify sources of feedback. Develop an appropriate test of the prototype. Conduct the test, collect and compile data, evaluate data, and decide on changes. Iterate the prototype or abandon the design idea.
Making	Identify and use appropriate tools, technologies, and materials for production. Make a plan for production that includes key stages, and carry it out, making changes as needed. Use materials in ways that minimize waste.	Identify and use appropriate tools, technologies, materials, and processes for production. Make a step-by-step plan for production and carry it out, making changes as needed. Use materials in ways that minimize waste.

Sharing	Decide on how and with whom to share their product. Demonstrate their product and describe their process, using appropriate terminology and providing reasons for their selected solution and modifications. Evaluate their product against their criteria and explain how it contributes to the individual, family, community, and/or environment. Reflect on their design thinking and processes, and evaluate their ability to work effectively both as individuals and collaboratively in a group, including their ability to share and maintain an efficient cooperative work space. Identify new design issues.	Decide on how and with whom to share their product and processes. Demonstrate their product to potential users, providing a rationale for the selected solution, modifications, and procedures, using appropriate terminology. Critically evaluate the success of their product, and explain how their design ideas contribute to the individual, family, community, and/or environment. Critically reflect on their design thinking and processes, and evaluate their ability to work effectively both as individuals and collaboratively in a group, including their ability to share and maintain an efficient cooperative work space. Identify new design issues.

Content

Finally, here are the computational thinking and coding content learning standards from Grades 6 to 9 supported by the student learning resources.

	Computational Thinking and Coding Content Learning Standards
Grades 6 to 7	· Simple algorithms that reflect computational thinking · Visual representations of problems and data · Visual programming
Grade 8	· Software programs as specific and sequential instructions with algorithms that can be reliably repeated by others · Debugging algorithms and programs by breaking problems down into a series of sub-problems
Grade 9	· Text-based coding

Computational Thinking in the BC Curriculum

In the redesigned BC Curriculum, computational thinking is part of the content learning standards for ADST from Grades 6 to 8. In Grades 6 to 7, computational thinking includes simple algorithms that reflect computational thinking, and visual representations of problems and data, and in Grade 8, computational thinking involves debugging algorithms by breaking problems down into a series of sub-problems.

While these computational thinking concepts are part of ADST, they are also evident in other learning areas across the BC curriculum. For example, in Arts 6 and 7, combining notations in music and dance to form sequences is an example of creating simple algorithms.

The tables below provide examples of Grade 6 to 8 learning standards from other subject areas that correspond to ADST learning standards.

Grade 6 Learning Standards

Subject	Learning Standard	Link to ADST Learning Standard
Arts	Notation in music and dance to represent sounds, ideas, movement, elements, and actions	Notations combined in a sequence is an example of a simple algorithm.
English Language Arts	Language features, structures, and conventions; specifically paragraphing	Developing paragraphs that are characterized by unity, development, and coherence is analogous to developing simple algorithms.
Mathematics	Model mathematics in contextualized experiences	Acting it out; using concrete materials (e.g., manipulatives); drawing pictures, diagrams, tables and graphs create visual representations of data.
	Order of operations with whole numbers	Order of operations is a simple algorithm for evaluating expressions.

Physical and Health Education	How to participate in different types of physical activities, including individual and dual activities, rhythmic activities, and games	Games usually involve rules that are simple algorithms.
Science	Construct and use a variety of methods, including tables, graphs, and digital technologies, as appropriate, to represent patterns or relationships in data.	Representing patterns or relationships creates visual representations of data.
	The basic structures and functions of body systems: • excretory • reproductive • hormonal • nervous	Functions of body systems follow simple algorithms.
Social Studies	Use Social Studies inquiry processes and skills to - ask questions; gather, interpret, and analyze ideas; and communicate findings and decisions	Locate and map continents, oceans, and seas using simple grids, scales, and legends; and represent the same information in two or more graphic forms (e.g., graphs, tables, thematic maps) are examples of creating visual representations of problems and data.
	Sequence objects, images, or events, and recognize the positive and negative aspects of continuities and changes in the past and present (continuity and change)	Sequencing is fundamental to creating simple algorithms.

Grade 7 Learning Standards

Subject	Learning Standard	Link to ADST Learning Standard
Arts	Notation in music and dance to represent sounds, ideas, movement, elements, and actions	Notations combined in a sequence is an example of a simple algorithm.
English Language Arts	Language features, structures, and conventions; specifically paragraphing	Developing paragraphs that are characterized by unity, development, and coherence is analogous developing simple algorithms.
Mathematics	Model mathematics in contextualized experiences	Acting it out; using concrete materials (e.g., manipulatives); drawing pictures, diagrams, tables and graphs create visual representations of data.
	Two-step equations with whole-number coefficients, constants, and solutions	The process for solving two-step equations is a simple algorithm.
Physical and Health Education	How to participate in different types of physical activities, including individual and dual activities, rhythmic activities, and games	Games usually involve rules that are simple algorithms.
Science	Collaboratively plan a range of investigation types, including fieldwork and experiments, to answer their questions or solve problems they have identified	Planning an investigation requires determining a sequence of steps similar to developing a simple algorithm.
	Construct and use a variety of methods, including tables, graphs, and digital technologies, as appropriate, to represent patterns or relationships in data.	Representing patterns or relationships creates visual representations of data.

Social Studies	Use Social Studies inquiry processes and skills to - ask questions; gather, interpret, and analyze ideas; and communicate findings and decisions	Demonstrate an ability to interpret scales and legends in graphs, tables, and maps (e.g., climograph, topographical map, pie chart); and select an appropriate graphic form of communication for a specific purpose requires visual representations of problems and data.

Grade 8 Learning Standards

Subject	Learning Standard	Link to ADST Learning Standard
Mathematics	Apply multiple strategies to solve problems in both abstract and contextualized situations; and develop, demonstrate, and apply mathematical understanding through play, inquiry, and problem solving	Solving mathematical problems often involves breaking the problems down into a series of sub-problems similar to debugging.
Science	Identify possible sources of error and suggest improvements to their investigation methods	Identifying possible sources of error is ultimately debugging and requires breaking their investigation methods down into smaller steps.

Why Computational Thinking?

One major objective of the redesigned BC Curriculum is to build students' core competencies in thinking, communication, and personal and social abilities in order to help students engage in deeper learning as well as life-long learning. In particular, creative thinking and critical thinking are core competencies that allow students to generate new ideas that are valuable to others and make judgements based on reasoning.

Computational thinking complements creative thinking and critical thinking. It allows students to solve complex problems. It involves looking at problems from different perspectives to develop potential solutions. When engaged in computational thinking, students may exercise one or more of the following strategies:

Strategy	Definition
Decomposition	Breaking something down into smaller pieces
Pattern recognition	Finding similarities between things
Abstraction	Removing unnecessary details
Algorithms	Sequencing of events

Computational thinking strategies are useful in all areas of learning, but they manifest themselves differently in each discipline. On one hand, students may use decomposition to break down a math problem into smaller pieces so that it's easier to solve. On the other hand, they may use decomposition to break down the elements of a story or text to gain deeper understanding of different forms and genres.

In programming or coding, computational thinking allows students to solve problems such that the solutions can be carried out by computers. It allows students to not only use, but also take control of technology to express their ideas and solutions. Furthermore, the coding process is highly iterative. Students will inevitably make mistakes when they write their code; however, they are able to quickly see the results, make changes, and iterate until they have their desired effect. Coding provides a safe environment to fail, and ultimately, builds resilience.

Approach to Training Workshops

Presenting Strategies

For the most part, participants in workshops are going to fall into two distinct groups. The first group will embrace computational thinking because they already have a solid familiarity with technology. The second group is more likely to express some skepticism and ambivalence.

Consistently, it has been demonstrated that the best way to bridge the gap between the facilitator and the second group is to present a conceptual map that feels familiar to teachers who may not consider themselves skilled at computation. Presenting computational thinking strategies in the context of how they already exist in everyday life helps build confidence and openness to how humans naturally compute solutions to problems every day. Computational thinking strategies are strengthened by the curricular competencies. The competencies are often easier to embrace because these are, for the most part, terms we already use.

To many people computational thinking strategies sound like jargon. It is best to address this directly, and let participants know that these kinds of words often trigger insecurities, or what is known to psychologists as "impostor syndrome," a feeling of being overwhelmed by new information that makes everyone less receptive to new ideas. Impostor Syndrome instills a sense that the participant isn't qualified or competent enough to accomplish the goal presented to them. But in truth, we put computational thinking strategies into practice often in our roles as educators, but also just as human beings.

Algorithms:

The sequence of events that lead from problem to solution. We have these in our everyday lives. Recipes, to-do lists, instructions to assemble furniture, and knitting patterns. These are all examples of algorithms, a sequence of events that, if followed, will lead from problem to solution reliably.

Pattern Recognition:

This is how we recognize faces, landmarks, and Wheel of Fortune clues. Our mind's ability to understand a pattern in data allows us to build algorithms that provide insights.

Decomposition:

Decomposition is the process of breaking down a complicated process into its component parts. Cleaning a messy room one section at a time or in timed increments is one example that might resonate well with middle schoolers. Quite often, this is how we create algorithms. And equally often, the process of breaking down a process leads to pattern recognition.

Abstraction:

We use abstraction when we allow one thing to represent another. In Algebra, we are all familiar with the concept of 'x' being an arbitrary value; but further from that, we know that red means stop, green means go, and a sale at the electronics store means we need to put money on our credit card. We have abstracted the meanings of these events and symbols so that we understand that they represent another series of actions. We abstract when we present the idea of a thing with a few well-chosen terms or symbols.

Linking Strategies to Curricular Competencies

By bridging the gap between the ADST curriculum and everyday practices, a contract is created between facilitator and participant to find success in the process of learning the new material. Computational strategies are strengthened and assimilated by being put into practice with the ADST curricular competencies. Some examples include:

Ideating and Testing
We are always trying new things. What happens if I do _____? We try a process, and see if it works. We are testing algorithms when we do this.

Sharing and Collaborating
The unsung hero of computational thinking, collaboration allows us to pair our perception of the problem and solution with others and get insights. Allowing ourselves to take input from others, particularly when they are recognizing patterns, decomposing the solution from their own experience, and using their own debugging and experimentation practices, allows us to come to a solution far more quickly.

Prototyping, Defining and Planning
We have all heard the saying 'proper planning prevents poor performance'. Planning means ensuring that the choices made in the building of an algorithm make sense, and the abstractions and experimentation techniques are all solid, and ultimately predict what the outcome will be based on the decisions. This is how we approach problem-solving in our lives each and every day, designing and defining solutions for ourselves and others.

Making
Once we have put computational thinking into practice, we gain not only the ability to solve problems, but new ways to express ourselves, whether it's using algorithms to tell stories, using abstraction to make animation, or logic to make games.

Big Ideas

By putting strategies into practice, the big ideas are naturally seeded. Ultimately a main goal of developing good computational thinking skills is to facilitate the learning process. Once again:

Big Ideas	Grade 6 to 8	Grade 9
Applied Design	Design can be responsive to identified needs.	Social, ethical, and sustainability considerations impact design.
Applied Skills	Complex tasks require the acquisition of additional skills.	Complex tasks require the sequencing of skills.
Applied Technologies	Complex tasks may require multiple tools and technologies.	Complex tasks require different technologies and tools at different stages.

Application

It is highly recommended that the concept of 'code' be addressed separately from 'computational thinking'. Computational thinking is only a method for solving a problem, whereas code is how we tell computers what to do. They are not the same thing and require different approaches. Code is only one aspect of the ADST curriculum, and is not introduced until grade 6, through simple visual programming. Whereas students and teachers alike can benefit from the introduction of computational thinking practices throughout the curriculum, in any grade or subject.

Workshop Itinerary Proposal

Here is a proposed outline and itinerary for doing a Computational Thinking (CT) workshop in your respective schools and districts. One of the approaches that has proven the most effective is getting participants to be hands-on with the material, following it up with discussion and reflection. The proposal below is for a 4-hour workshop; however, should you have more time available there are some notes below for extending the material.

We recommend starting workshops with introductions and a "Shape of the Day" outline so that attendees have expectations set appropriately. Each session should end with an introduction to the Discourse resource.

Hour 1

Ping-Pong Rescue activity from the first unit of the Student Learning Module. (60 - 75 minutes)

1. Divide the attendees up into groups and have them perform the activity. This will serve as a good icebreaker and set the mood for the rest of the session. (45 minutes)
2. Follow up with a "hook", connecting the lessons learned and competencies explored in the Ping-Pong activity to the ADST curriculum. (15 - 30 minutes)

Hour 2

Discussion (60 minutes)

1. How does CT exist in everyday life / classroom settings ALREADY. (15 - 20 minutes)
2. For each content component of CT (Decomposition, Abstraction, Pattern Recognition, and Sequencing), get teachers to work with their groups (one by one) to make connections to their current practice. (30 - 45 minutes)

Hour 3

Module Overview (60 minutes)

1. Module 1 (25 minutes)
2. Module 2 (25 minutes)
3. Emphasis on multi-disciplinary/cross-curricular application/implementation (10 minutes)

Hour 4

Reflection (30 minutes)

1. Reflecting on integration of computational thinking into current curriculum and teaching practice
2. If time allows, opportunity to reflect after each module overview session, with additional reflection around cross-curricular use

District-specific or school-specific Q&A (30 minutes)

1. Assessment expectations
2. Prep expectations for new curriculum
3. Addressing concerns over new curriculum potentially disrupting previous lesson allotments

Extended Sessions

Here are some options for extending the session if time allows:

1. Use the Jeopardy resource that was introduced at the Provincial Workshops.
2. Have participants explore either the Kahoot! or Twine activities from the Student Learning Module.
3. Get participants to reflect on rubrics for assessing computational thinking.
4. Explore concepts of multi-disciplinary and cross curricular implementation of the CT content.

Further Resources

Learn to Code, Code to Learn, Mitchel Resnick, EdSurge
https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn

Let's Teach Kids to Code, Mitchel Resnick, TED
http://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code?language=en

Education Reform, Brian Aspinall, TEDxChathamKent
https://youtu.be/ngeZPU35zm4

Hacking the Classroom, Brian Aspinall, TEDxKitchenerED
https://youtu.be/UyxfPnO5lgk

New Frameworks for Studying and Assessing the Development of Computational Thinking,
Karen Brennan and Mitchel Resnick, MIT Media Lab
http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

The Scratch Ed Creative Computing Curriculum Guide has been created at Harvard and offers guidance and lesson plans for teaching Scratch in the classroom:
http://scratch.mit.edu

The mission of Code Club is to give every student and child the opportunity to learn code. A bank of step-by-step projects in Scratch, HTML and Python facilitates the creation of after school clubs for an increased network and support.
http://codeclub.ca

The Hour of Code initiative has sparked interest in coding internationally. This section of the site is dedicated to educators at all grade levels who are interested in techniques and resources for bringing computational thinking and computer science to their classrooms.
http://code.org/educate


BRITISH
COLUMBIA