DIGITAL TECHNOLOGY AND INNOVATIONS IN THE CHANGING WORLD, ICD2O COURSE PLAN 2026

The Ontario Science Centre and the Association for Computer Studies Educators, with support from the Ontario Ministry of Education, are pleased to present this course plan for Digital Technology and Innovations in the Changing World (ICD2O), designed to support educators in their delivery of this program.

Course Plan Introduction

Vision and Goals

"The vision for this course is for students to develop the knowledge and skills related to digital technology and computer programming that will support them in contributing to and leading the global economic, scientific, and societal innovations of tomorrow. Digital technologies play a major part in all aspects of our lives, and the course supports students in understanding how those digital technologies work and how they can be used and developed for the common good. The course enables students to explore how to use computing and critical thinking to address issues that are meaningful to them and their communities, and how to move from being consumers of digital technologies to becoming empowered creators" (Ontario Ministry of Education, 2023). See Computer Studies: Digital Technologies and Innovations in the Changing World, Grade 10, Open (ICD2O) - https://www.dcp.edu.gov.on.ca/en/curriculum/computer-studies/courses/icd2o/course-intro.

Digital Technology and Innovations in the Changing World — Scope and Overview

Digital Technology and Innovations in the Changing World, Grade 10 (ICD2O) is:

- an introduction to computational thinking and digital innovation
- focused on concepts and logic rather than specific rules of a language
- designed to build confidence and foundational fluency in digital technologies
- intended for students with a wide range of prior programming experience

This course is not:

- a university-preparatory computer science course
- a course requiring advanced mathematics
- a course requiring teachers to be an artificial intelligence (AI) or cybersecurity expert

Course Plan — Scope and Overview

This course plan outlines a possible sequence of learning, but every educator should adopt their own personal approach based on the lived experiences of their students and their own experience and knowledge.

Some key considerations regarding the design of this course plan include:

- **Comprehensive Coverage:** All specific expectations are addressed within this plan.
- **Recurring Themes:** Critical expectations like *A2.1 investigate current social, cultural, economic, environmental, and ethical issues related to digital technology that have personal, local, and global impacts, taking various perspectives into account* are revisited throughout the course.
- **Scaffolded Learning:** The sequence of learning in the plan prioritizes prerequisite knowledge as it addresses the cumulative nature of programming and digital literacy.
- **Flexibility in Tools:** While educators may choose their preferred programming languages and tools, selections should align with **Universal Design for Learning (UDL)** principles to ensure an equitable experience for all learners. This course may be implemented using a variety of programming environments (for example, Python, JavaScript or other introductory programming platforms).

The following is an approximate pacing guide for a full semester course (110 hours total):

- **Unit 1 — Building Digital Foundations: 4–6 weeks (35 hours)**
- **Unit 2 — Living in a Connected World: 4–6 weeks (35 hours)**
- **Unit 3 — Shaping the Future: 4–5 weeks (30 hours)**
- **Culminating Activities and Reflection: 1–2 weeks (10 hours)**

Backward Design Process

The backward design process involves three phases:

1. Identify desired results
2. Determine assessment evidence (see Assessment Philosophy and Framework)
3. Plan learning experiences and instruction (see Universal Design for Learning Considerations)

We will examine the **enduring understandings** and **essential questions** central to computer science and digital technology.

Enduring Understanding	Essential Questions
Data can be represented and manipulated using algorithms by writing software using a programming language.	What type of data can be represented within a digital computer? How can I ensure the algorithms are implemented correctly?
Computers can be used to help humans make decisions.	How do computers provide recommendations? Are humans part of the decision-making process?
Technology carries the values, biases and priorities of its creators.	What are the ethical implications of emerging technology? How can we identify bias in technology solutions?
Computers do exactly what you tell them, not what you want them to do.	What are the important elements of programming languages that must be learned to direct a computer to solve a problem? What are computational thinking (CT) skills and how can they be applied to plan and develop software?
Writing software involves communicating precisely through a programming language and communicating clearly with other developers through embedded documentation and readable code.	Why is "working code" not necessarily "good code"? What are some industry-standard processes associated with writing code so others can understand and possibly extend its functionality?
Problems are rarely unique. By identifying trends and patterns within data or logic, we can apply previously successful solutions to new challenges.	When I recognize patterns, how can I use programming constructs like loops, data structures or subprograms to solve the problem using code?
Artificial intelligence is a tool that requires constant human oversight and ethical calibration.	How can I better understand when I'm interacting with an AI system? What data is used by the AI system and how has it been validated?

The preceding table identifies some of the enduring understandings and essential questions that could be considered as you develop your assessment evidence and plan learning experiences and instruction.

Assessment Philosophy and Framework

While this course plan focuses primarily on sequencing and instructional design, effective implementation also requires clarity around assessment practices. In alignment with a backward design framework and Growing Success policy, educators are encouraged to consider how assessment evidence will demonstrate students' understanding of computational thinking, programming concepts and the societal implications of digital technologies.

Assessment in ICD2O should:

- be ongoing and primarily formative throughout each unit of learning
- emphasize both process and product
- include opportunities for observation and conversations along with products
- provide multiple opportunities for student reflection and revision
- balance technical skill development with ethical and societal analysis
- align with Universal Design for Learning principles

Given the three thematic units of the course, educators may consider structuring assessment within each unit.

Unit-Based Assessment Structure

Each unit may include:

- ongoing formative tasks (for example, debugging exercises as entry or exit tickets, algorithm design tasks, case study analysis and research checkpoints)
- one performance-based culminating task that integrates programming skills, systems understanding and social or ethical connections
- a reflective component that encourages students to evaluate their design decisions, computational thinking strategies and the broader impacts of their work

Culminating tasks at the end of each unit may include:

- development of an interactive software application (Unit 1)
- a modular program addressing a data or cybersecurity challenge (Unit 2)
- an ethical innovation project integrating classical computing and/or AI tools (Unit 3)

Assessment categories must reflect the expectations of *Growing Success*, including Knowledge, Thinking, Communication and Application. Given the hands-on, project-based nature of the course, Thinking and Application may be emphasized within performance tasks. The Achievement Chart for the Grade 10 Computer Studies Course (ICD2O) (source: https://www.dcp.edu.gov.on.ca/en/curriculum/computer-studies/courses/icd2o/assessment-and-evaluation) contains many examples that could be incorporated into an assessment plan across the four categories of knowledge and skills.

Educators may also consider implementing a digital portfolio approach to support documentation of growth over time, including code artifacts, debugging reflections, ethical analyses and peer feedback.

This framework is intended as guidance rather than prescription, allowing educators flexibility in tool selection, task design and local contextualization.

Universal Design for Learning (UDL) Considerations

As educators explore the proposed sequencing and learning described in this course plan it is important to consider applying UDL to provide multiple entry points for students. The course tends to have a wide range of students as some have been learning to program for many years while for others the process of designing and writing code might be quite new.

Reference: Instructional approaches from Digital Technology and Innovations in the Changing World: https://www.dcp.edu.gov.on.ca/en/curriculum/computer-studies/courses/icd2o/program-planning-ccil#instructional-approaches

UDL provides a framework for educators to consider all students, including providing multiple means of engagement, representation and expression.

Engagement

Provide students with choices as they explore technology or develop software. Some students may want to design programs that create art and others might be interested in creating interactive games. Use a variety of instructional strategies including direct instruction for new content and collaborative learning (pair programming) as students work together to solve problems and explore emerging technologies. Clearly communicate expectations and guide students to reflect on their understanding through formative assessments and descriptive feedback. The ICD2O course includes a focus on explaining computational artifacts to others. This could be practised throughout the course and the culminating projects as students are often very proud of their projects and enjoy sharing them with others in a final showcase or gallery walk event.

Representation

Educators can use analogies and manipulatives to aid in student understanding of abstract concepts commonly encountered in Computer Studies. An example of a common misconception is the use of an assignment operator (=) and the equality operator (==). Providing visual planning examples such as flowcharts or pseudocode can also help students make connections between computational thinking (CT) skills of decomposition and algorithms prior to writing and testing code. Live coding is another excellent active learning technique where educators "think through" the design and implementation in a whole class discussion, interacting with students throughout the process. Live coding can include intentional debugging moments

that model resilience and problem-solving strategies. Live tracing code is another valuable technique. In this approach, the educator (or students) step through a program line by line, predicting and tracking what happens at each iteration. This makes program flow and state changes visible and helps students develop stronger mental models of how code executes by breaking it down step by step. Live tracing can be reinforced through brief entrance or exit tickets in which students trace a short script independently. Educators can set clear learning goals and success criteria for students as this clarifies expectations and reduces student anxiety. When assessment and evaluation criteria are co-developed with the learners it provides an opportunity for mutual agreement of a variety of methods for them to demonstrate their skills and understanding.

Action and Expression
Students can be provided with multiple options to communicate their understanding through written, visual or oral methods. Educators can facilitate discussions of problem-solving strategies and provide worked examples as students develop their skills. Graphic organizers are also useful for some students as they connect new concepts with previous topics.

Learning Skills and Work Habits

Learning skills and work habits are developed and assessed throughout the course and are reported separately from academic achievement.

In ICD2O, these learning skills are closely connected to computational thinking, programming practice and collaborative project work. Students will develop and demonstrate responsibility, organization, independent work, collaboration, initiative and self-regulation as they plan algorithms, manage digital files, debug code, work with peers, seek feedback and reflect on their learning.

Course Plan Organizational Structure

This course plan is organized into the following three **units**, each representing a major thematic phase of learning.

1. Building Digital Foundations
2. Living in a Connected World
3. Shaping the Future

Within each unit, learning is further organized into the following **clusters**, designed to group related expectations and instructional focuses.

1. Exploring Technology
2. Designing Software
3. Connecting Innovations in Society

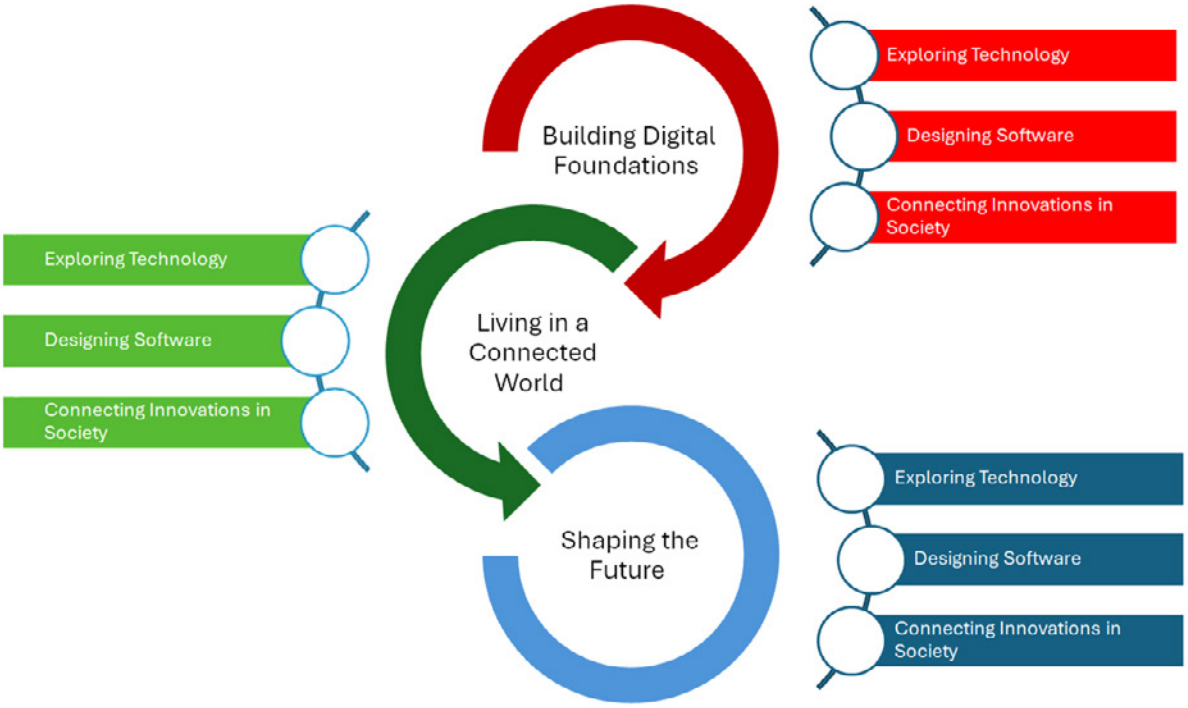
The structure can be summarized as:

Unit > Clusters > Expectations > Learning Activities and Assessment

This structure allows teachers to organize instruction around meaningful themes while maintaining clear alignment with curriculum expectations.

Each unit progresses from foundational understanding (Exploring Technology) to applied skills (Designing Software) and finally to reflection on societal impact (Connecting Innovations in Society).

The graphic below illustrates the relationship between units, clusters and curriculum expectations within the course structure.



Unit 1 — Building Digital Foundations

Suggested Duration: 4–6 weeks (35 hours)

Digital foundations include elements of hardware and software. Students will explore the core components of computing devices and then determine how they can become creators rather than just consumers of technology. Students may already have some coding experience through the integration of coding in the mathematics and science curriculum Grades 1–9. This unit will introduce computational thinking (CT) concepts and skills. It will apply them within the context of creating simple applications that involve the use of programming language fundamentals such as the use of variables to manage state and selection statements to make decisions. The impacts of technology and computation on various disciplines (for example, computational biology) will also be explored.

Exploring Technology

This cluster introduces students to the fundamental concepts and systems that support digital technologies, including hardware components, operating systems and the role of computing across disciplines.

Students may demonstrate learning through research briefs, technology comparison reports or short presentations analyzing computing systems.

Skills and Connections	Specific Expectations
Students will: • research the core hardware components of a computer to determine the usability and performance features of a computing system • discuss the selection criteria for recommending a computing system for a diverse set of users • explore types of processors used in modern computing devices • examine the features of operating systems and explore how each of these features impacts security, usability and performance • perform a thorough analysis of the software systems that they use on a frequent basis; the analysis should consider the type of software and its features and functionality • make recommendations for hardware and software for specific scenarios (for example, a scientist performing research in a remote location) • use file management techniques, including cloud storage, to support collaborative research tasks	B1.1, B1.2, B1.3

Designing Software

This cluster focuses on developing computational thinking skills and applying them to the design and implementation of simple software programs.

Assessment opportunities may include algorithm design tasks, debugging challenges or the development of small interactive programs.

Skills and Connections	Specific Expectations
Students will: • explore and discuss computational thinking (CT) concepts including decomposition, pattern recognition, abstraction and algorithms using computational challenges available within the computer science community[1] • apply CT concepts to create simple software solutions using a text-based coding language • design and communicate algorithms using planning tools such as flowcharts or pseudocode • create software that accepts user input, makes decisions based on provided data and produces desired output • investigate teacher-provided programs to identify errors and modify these programs to meet the requirements provided • test programs using a strategy to ensure algorithm correctness • apply problem-solving strategies in scenarios that require the use of math operators and relational/conditional expressions	A1.1, A1.2, A1.3, C1.1, C1.2, C1.3, C1.4, C1.5, C2.1, C2.3, C2.4, C2.5, C2.6, C2.7

[1] https://cemc.uwaterloo.ca/contests/bcc

Connecting Innovations in Society

This cluster encourages students to explore how computing technologies impact society, culture and other disciplines.

Students may demonstrate understanding through reflective writing, case study analysis or presentations examining the societal impacts of technology.

Skills and Connections	Specific Expectations
Students will: • explore how computing systems are integrated into other disciplines (for example, science, mathematics, art, music and business) • investigate how digital technologies are being used to protect and teach Indigenous languages, manage resources and document oral histories • investigate contributions to digital technology and computing innovations within these disciplines • reflect on social, cultural, economic, environmental and ethical issues when considering impacts within these disciplines	A2.1, A2.3, A3.1

Unit 2 — Living in a Connected World

Suggested Duration: 4–6 weeks (35 hours)

Students live in a fully connected world. In this unit, they will examine the benefits and limitations of connected systems while exploring how to stay safe online. Designing interactive software in this unit involves working with more complex techniques and data structures to meet the needs of a variety of different users in different contexts. Students will start to explore more complex data structures such as lists and files. Connections will focus on examining the impacts of technology and computation on various industries. The following clusters build students' understanding of connected systems, expand their programming skills and examine the impact of digital technologies across industries.

Exploring Technology

This cluster focuses on understanding how devices connect, how networks function and how cybersecurity practices protect digital systems.

Students may demonstrate learning through network diagrams, cybersecurity case analyses or recommendations for safe online practices.

Skills and Connections	Specific Expectations
Students will: • explore the techniques used to connect devices using wired and wireless methods • consider the performance and integrity of these connected systems • explore cybersecurity methods that support confidentiality (for example, authentication and encryption), integrity (for example, checksums) and availability (for example, backups that reduce the impact of ransomware attacks) • identify the core elements of computer networks including hardware components and related protocols • apply their understanding of network protocols to determine which websites have implemented encryption protocols to maintain safety • use effective research practices to investigate current cybersecurity vulnerabilities and identify how to protect users and computing systems from threat actors • develop a set of recommendations for configuring home networking technologies and managing personal habits to stay safe online	A2.2, B2.1, B2.2, B2.3, B3.1, B3.2, B4.1, B4.2, B4.3

Designing Software

This cluster extends students' programming skills by introducing modular design, data handling and more complex program structures.

Assessment opportunities may include modular coding tasks, data visualization activities or debugging exercises involving code tracing of loops.

Skills and Connections	Specific Expectations
Students will: • identify patterns within algorithms and design solutions using conditional or counted loop structures • decompose larger programs into independent modules or functions that can be tested separately • explore and use programming language libraries to visualize data and identify trends within the data	A1.1, A1.2, A1.3, C1.5, C2.2, C2.4, C3.3, C3.4

Connecting Innovations in Society

This cluster encourages students to examine how digital technologies influence industries and the broader economy.

Students may demonstrate understanding through industry case studies, comparative analysis or short reflections on the impacts of digital technologies.

Skills and Connections	Specific Expectations
Students will: • explore how computing systems and digital technologies are impacting different industries, including skilled trades and the growing use of artificial intelligence systems • investigate contributions to digital technology and computing innovations within these industries • reflect on the social, cultural, economic, environmental and ethical issues when considering the impacts within these industries	A2.1, A2.3, A3.2

Unit 3 — Shaping the Future

Suggested Duration: 4–5 weeks (30 hours)

Students will build upon their digital foundation skills of computing systems, networks and software development to tackle innovations in digital technologies. They will investigate some of the most commonly used types of artificial intelligence and examine their possible benefits and limitations. Students will have opportunities to incorporate AI tools into their programs. At this level, exploration of AI concepts is conceptual rather than mathematical. The emphasis is on understanding how systems function and their societal implications, not implementing complex models from scratch. The following clusters support students in exploring emerging technologies, applying responsible innovation practices and examining the societal implications of AI and digital change.

Exploring Technology

This cluster introduces students to key concepts in artificial intelligence and machine learning while examining the ethical considerations of these technologies.

Students may demonstrate learning through case study analysis, research summaries or discussions evaluating the benefits, limitations and ethical implications of AI systems.

Skills and Connections	Specific Expectations
Students will: • identify everyday computing systems that use machine learning and other artificial intelligence techniques to make decisions then compare them with classical computing systems • explore how classification systems are created using machine learning methods that use labeled data and supervised learning techniques • discuss how reinforcement learning models can be used to improve decision making (for example, autonomous vehicles) • explore how associations and clusters of similar data can be identified using unsupervised learning techniques • explore the ethics associated with training machine learning models, including the environmental impacts of AI systems and the possibilities of building biased models that reinforce historical inequities • discuss how data sets are obtained and how humans tag data to support supervised learning AI models • research model cards or other documentation provided with AI models to better understand their limitations and accuracy	A2.4, B4.1, B4.3

Designing Software

This cluster allows students to apply programming skills to create applications that incorporate AI tools, user interface considerations and accessibility features.

Assessment opportunities may include interface design tasks, prototype development or reflections on fairness, bias and accessibility in digital tools.

Skills and Connections	Specific Expectations
Students will: • develop a software program that uses classical decision-making structures and determine if the decisions are considered fair and free of bias • use tools to build a custom supervised-learning AI model to help identify images and embed the model within a user-facing application • explore various user interface options for the applications they create • research accessibility features for applications and websites • examine assistive devices for visually impaired users • showcase and present software projects and extend their functionality based on user feedback	A1.1, A1.2, A1.3, A2.5, B2.3, C3.1, C3.2, C3.3, C3.4, C3.5

Connecting Innovations in Society

This cluster encourages students to examine how emerging technologies influence careers and society.

Students may demonstrate understanding through career research, ethical analyses or presentations on the societal impacts of emerging technologies.

Skills and Connections	Specific Expectations
Students will: • identify the economic and ethical impacts associated with the adoption of digital technology (and specifically AI systems, AI and other technologies) within the context of different careers • investigate contributions to digital technology and computing innovations within these careers • reflect on the social, cultural, economic, environmental and ethical issues when considering the impacts within these careers	A2.1, A2.3, A3.3

Taken together, these units support students in developing foundational digital literacy, computational thinking and an understanding of the broader impacts of digital innovation.

Culminating Activities and Reflection

Suggested Duration: 1–2 weeks (10 hours)

The final portion of the course may include a culminating activity and reflective process that provides students with an opportunity to synthesize learning across the strands. Students may complete an innovation project in which they design or refine a computational artifact, software application or innovation-focused solution that responds to a meaningful problem, user need or ethical question. The culminating task should provide opportunities for students to demonstrate learning from Strand A, Strand B and Strand C in an integrated way.

The culminating activity may also include a reflective component, presentation or conference in which students explain:

- the purpose and audience for their artifact
- the computational thinking strategies used in its development
- relevant design choices and debugging processes
- accessibility, fairness or ethical considerations
- how their understanding of digital technology and innovation has developed across the course