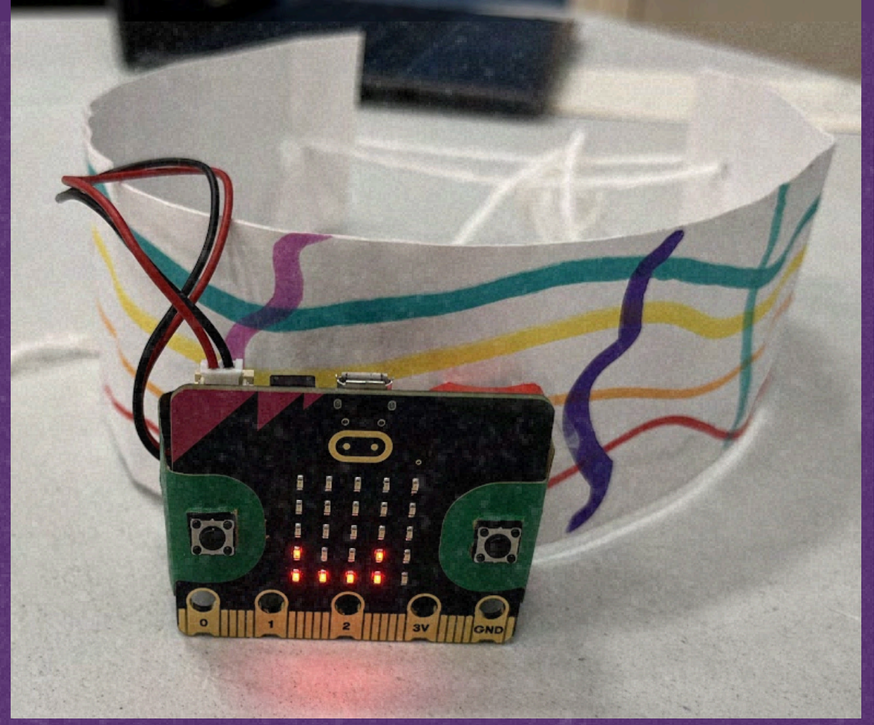
Welcome to our exciting journey into coding!

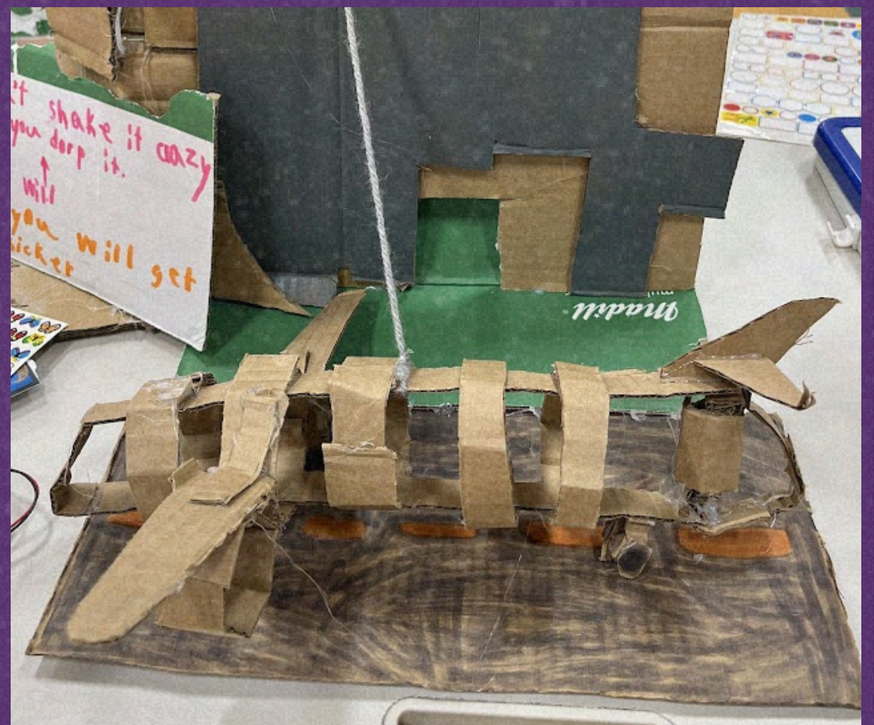
Coding the Core – Introducing Grades 4–6 Computer Science

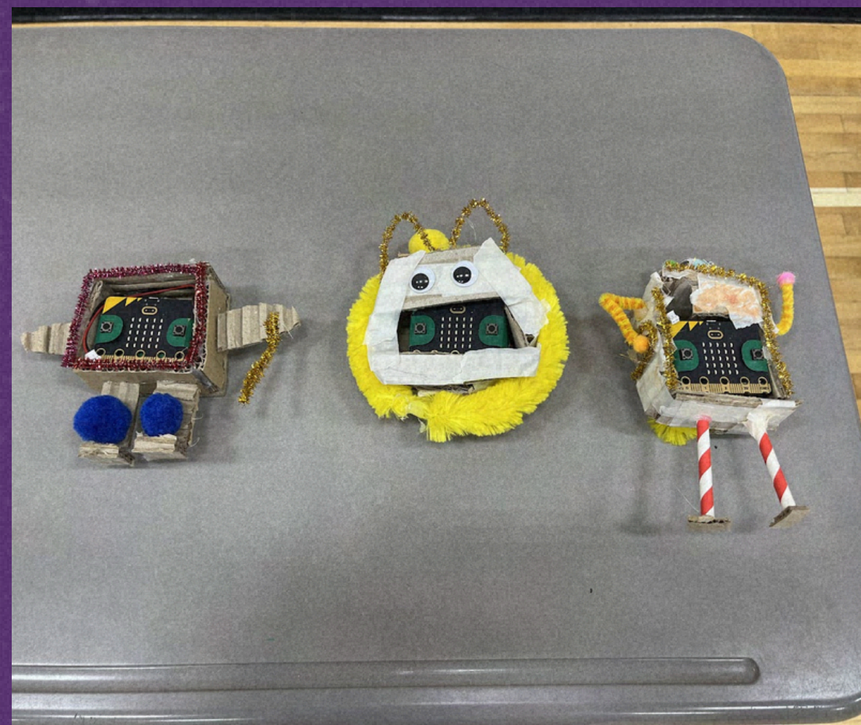
Angela Dearing

angela.dearing@wrps11.ca

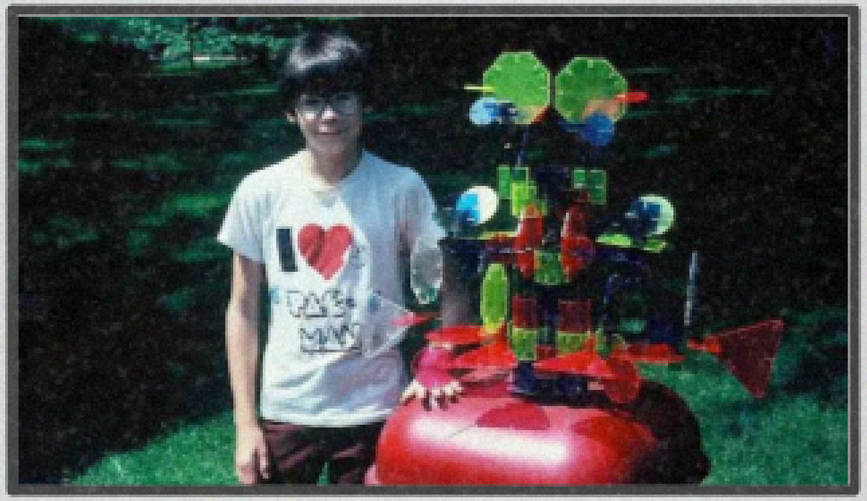

ROBO kitty
version 1
FEED
Hungry
Kitty
Please

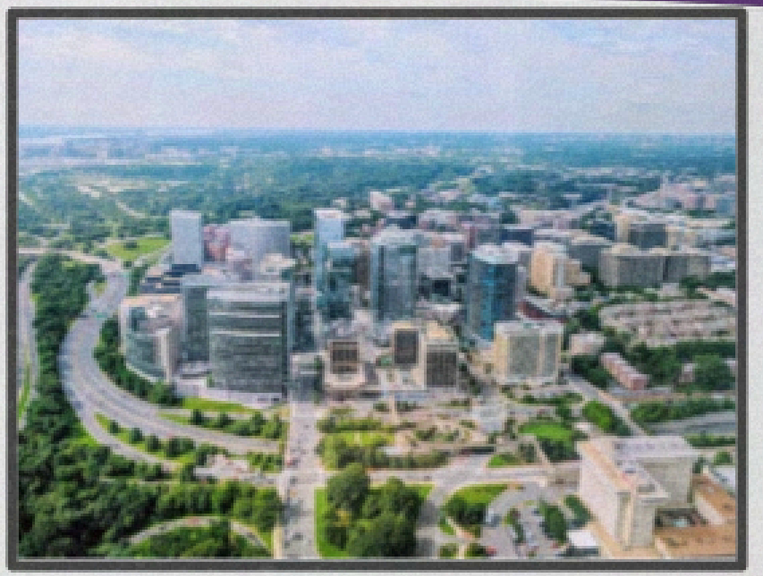




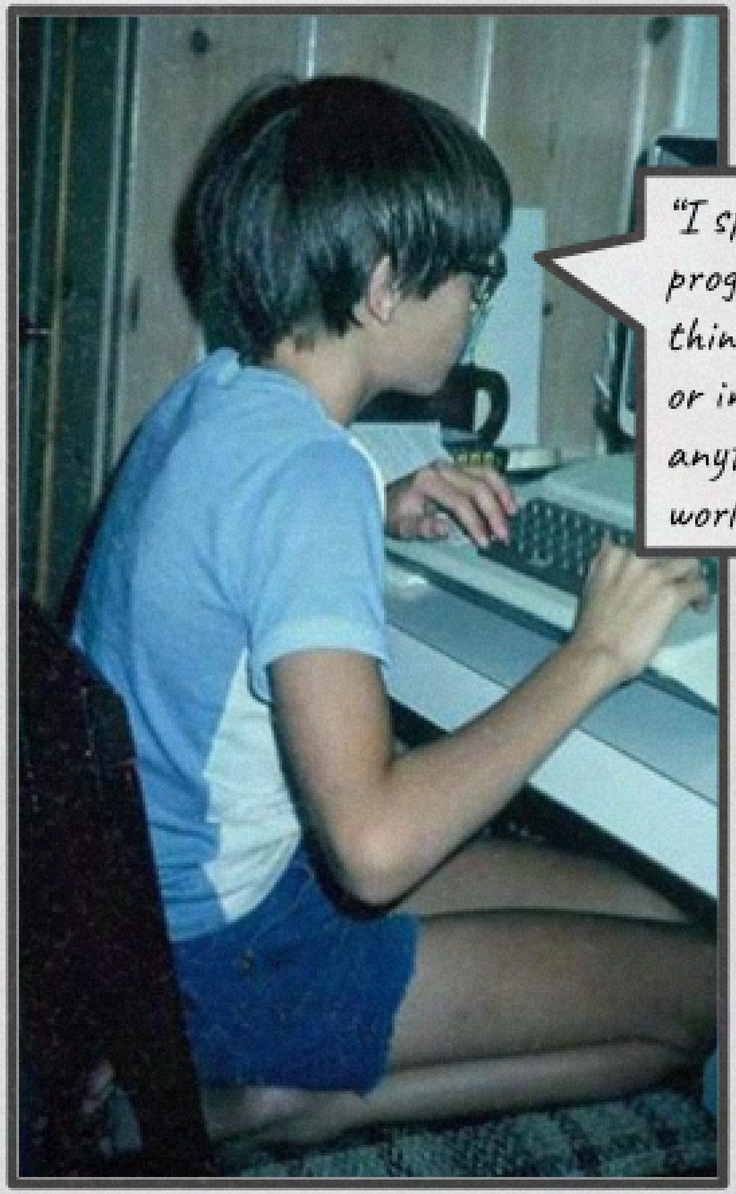


The Story of Timothy Sweeney





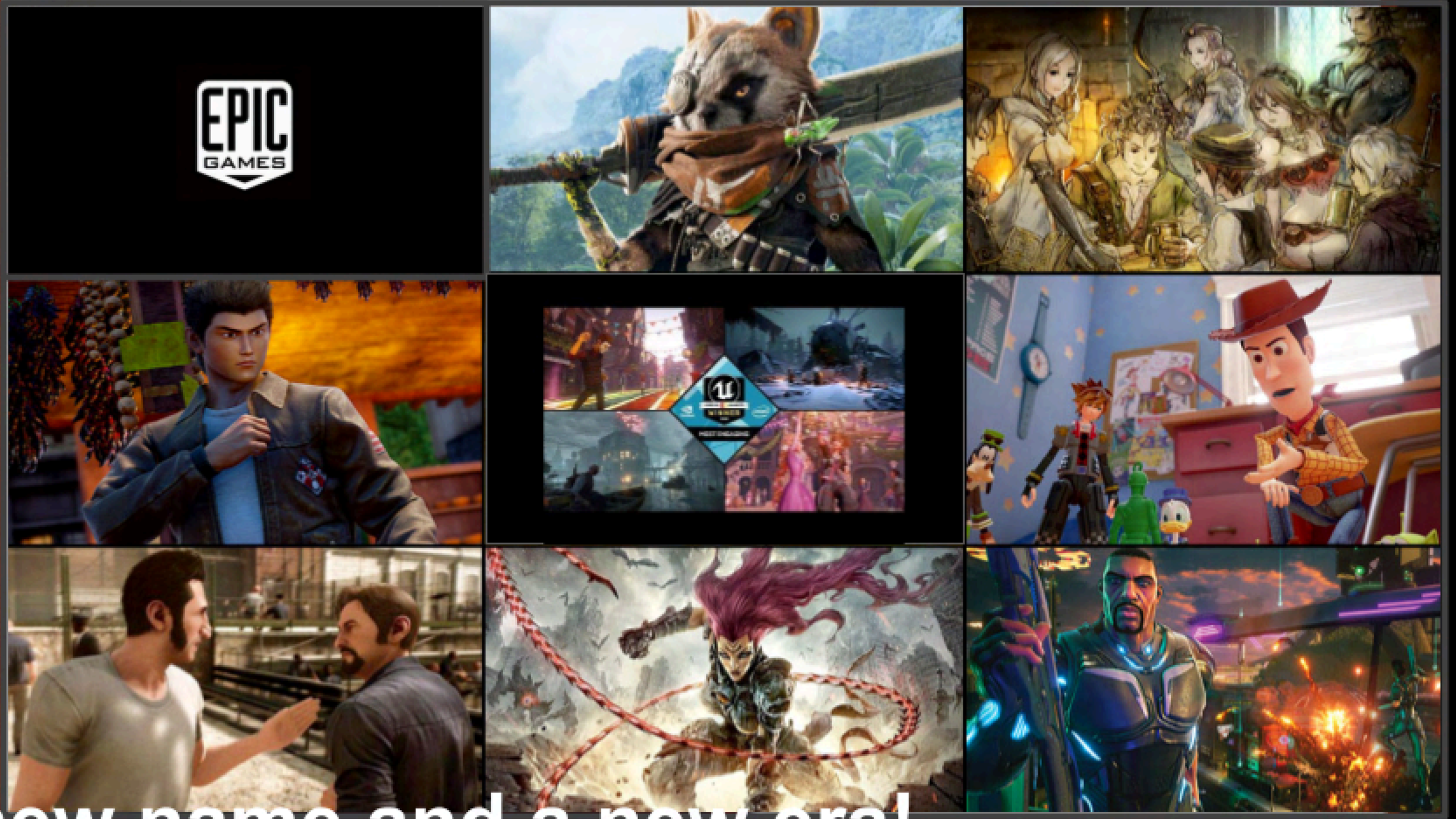
```c
/*
 * This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world!\n" to
 * standard output when invoked.
 */
void anyHello(void) {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!\n");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
```



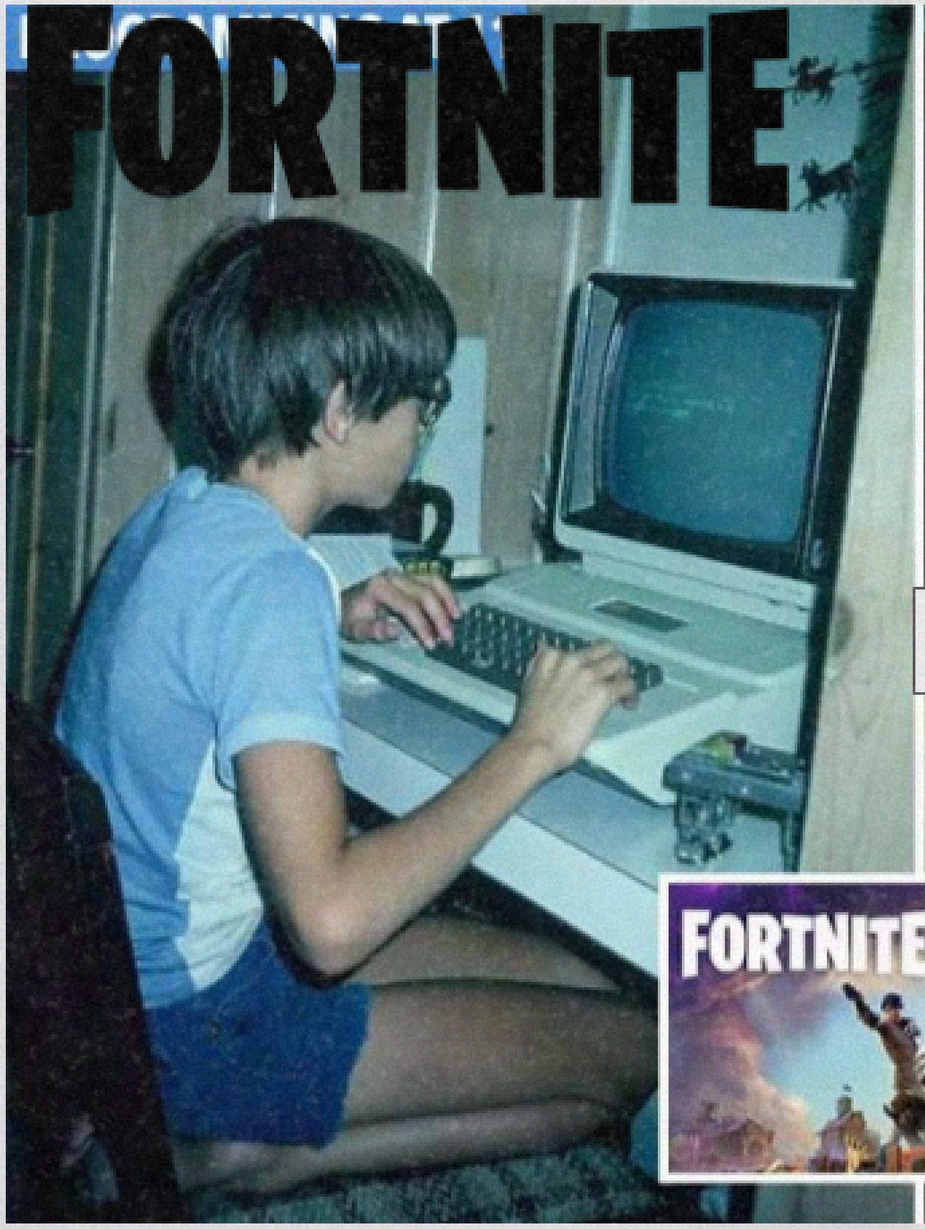
"I spent more time programming than I think I was sleeping or in school or doing anything else in the world."

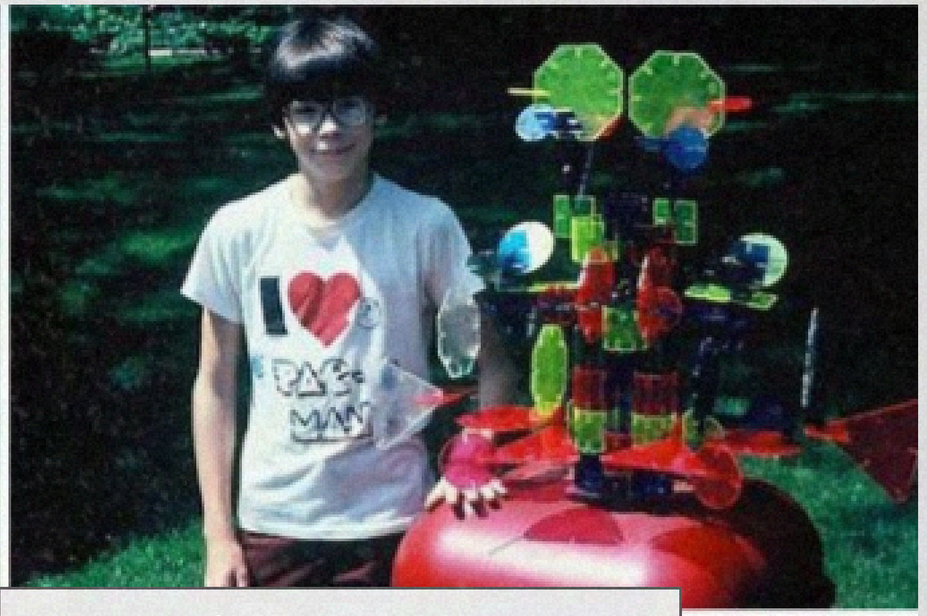

MARIO
000000
×00
WORLD
1-1
TIME
SUPER
MARIO BROS.
©1985 NINTENDO
1 PLAYER GAME
2 PLAYER GAME
TOP- 000000


Prison
Prison Cell
Exit
Time
Bridge
The Gap
ZZT
Time:136
Health:92
Ammo:222
Torches:9
Gems:60
Score:909
Keys:
T Torch
B Be quiet
H Help
Move
Shift Shoot
S Save game
P Pause
Q Quit



A new name and a new era!





Worth $55B US today


What will your
DREAM be?


CES
2025


BEST OF
CES
2025
ozlo


BEST OF
CES


Best
of CES

Problem:

The younger students need to learn about computational thinking and computer coding but their teachers don’t know how to teach it.

They have asked you to invent some new toys and games that will:

- ❏ teach students about computer programming/coding
- ❏ keep the students entertained
- ❏ make the students excited to learn about coding

Computer Science

Problem solving and scientific inquiry are developed through the knowledgeable application of creativity, design, and computational thinking.

Grade 4: Students examine and apply design processes to meet needs.

Grade 5: Students apply design processes when creating artifacts that can be used by a human or machine to address a need.

Grade 6: Students examine abstraction in relation to design and coding and describe impacts of technologies.

Knowledge	Understanding	Skills & Procedures
Design processes include • understanding the problem • forming ideas (ideating) • planning • creating • analyzing • testing • troubleshooting Feedback helps to ensure all needs are considered during the design process. An algorithm is a sequence of instructions. Artifacts are objects or products made by humans, machines, or computers through the process of design. Design can produce many artifacts, including • algorithms • models • prototypes • blueprints • programs • experiments • objects Design can deal with complex problems. Availability of materials and costs are considerations in design.	Design involves processes that can transform ideas into artifacts that meet needs.	Plan and create an artifact to meet a need. Provide feedback to others during the design process. Test an artifact to confirm that it meets intended needs. Collaborate to design an algorithm to solve a problem. Examine availability and cost of materials during design. 

Grade 4 Skills & Procedures

- Plan and create an artifact to meet a need.
- Provide feedback to others during the design process.
- Test an artifact to confirm that it meets intended needs.
- Collaborate to design an algorithm to solve a problem.
- Examine availability and cost of materials during design.

Knowledge	Understanding	Skills & Procedures
A computational artifact is anything created by a human using a computer, such as computer programs and code images audio video presentations web pages Design can be used to create algorithms and translate them into code. Code is any language that can be understood by and run on a computer. There are many ways to code, including using visual block-based languages. Visual block-based languages are a form of code in which prepared chunks of instructions are in drag-and-drop blocks that fit together like puzzle pieces to design a program. A computer cannot think for itself and must rely on code for all that it does. A loop is a repetition of instructions used in an algorithm.	Design can be used by humans or machines to meet needs.	Engage in the design process to create computational artifacts. Relate a block of code to an outcome or a behaviour. Explain what will happen when single or multiple blocks of code are executed. Translate a given algorithm to code using a visual block-based language. Design an algorithm that includes a loop and translate it into code. 

Knowledge	Understanding	Skills & Procedures
Design process can be influenced by various factors, including safety functionality usability reliability efficiency aesthetics Functionality is the quality of being useful to do the job for which something was designed. Usability is the degree of ease with which something can be used to achieve an outcome. Design processes that support the development of multiple iterations include enhancing refining Design can be improved through collaboration.	Design can better meet needs through the development of multiple iterations.	Discuss examples of designs that have been enhanced or refined to better meet needs. Evaluate an artifact based on various factors. Design an artifact to meet a need. Propose enhancements and refinements to an artifact in collaboration with others. Develop multiple iterations of an artifact. 

Grade 5 Skills & Procedures

- Engage in the design process to create computational artifacts.
- Relate a block of code to an outcome or a behaviour.
- Explain what will happen when single or multiple blocks of code are executed.
- Translate a given algorithm to code using a visual block-based language.
- Design an algorithm that includes a loop and translate it into code.
- Discuss examples of designs that have been enhanced or refined to better meet needs.
- Evaluate an artifact based on various factors.
- Design an artifact to meet a need.
- Propose enhancements and refinements to an artifact in collaboration with others.
- Develop multiple iterations of an artifact.

Knowledge	Understanding	Skills & Procedures
The process of abstraction includes determining what details to keep and what to ignore removing unnecessary details identifying important information generalizing patterns Information is data that is organized to be more useful. An abstraction is a simplified version of something complex. Abstractions can make daily life easier; e.g., simple controls on appliances light switches steering wheels apps Computational artifacts can be designed to address societal needs and wants; e.g., weather modelling communications automotive controls medical research apps Structures used in coding include sequences conditionals; e.g., if–then–else statements loops Sequence structures are ordered sets of instructions within code. Conditional structures are statements that tell computers to complete different actions based on different situations.	Abstraction is used in design and coding of computational artifacts to make problems easier to think about.	Apply abstraction during the design process. Identify examples of abstractions encountered in daily life. Discuss the role of design and coding in society. Use a visual block–based language to design code that includes relevant design structures. 

Knowledge	Understanding	Skills & Procedures
The use of computers, coding, and technology can have impacts that are personal social environmental economic Impacts of computers, coding, or technology may be intentional or unintentional.	Computers, coding, and technology can be used in ways that have positive or negative impacts.	Discuss how computers, coding, or technology have had impacts. Predict possible impacts of computers, coding, or technology.

Grade 6 Skills & Procedures

- Apply abstraction during the design process.
- Identify examples of abstractions encountered in daily life.
- Discuss the role of design and coding in society.
- Use a visual block-based language to design code that includes relevant design structures.
- Discuss how computers, coding, or technology have had impacts.
- Predict possible impacts of computers, coding, or technology.

Creativity

Design

Computational Thinking

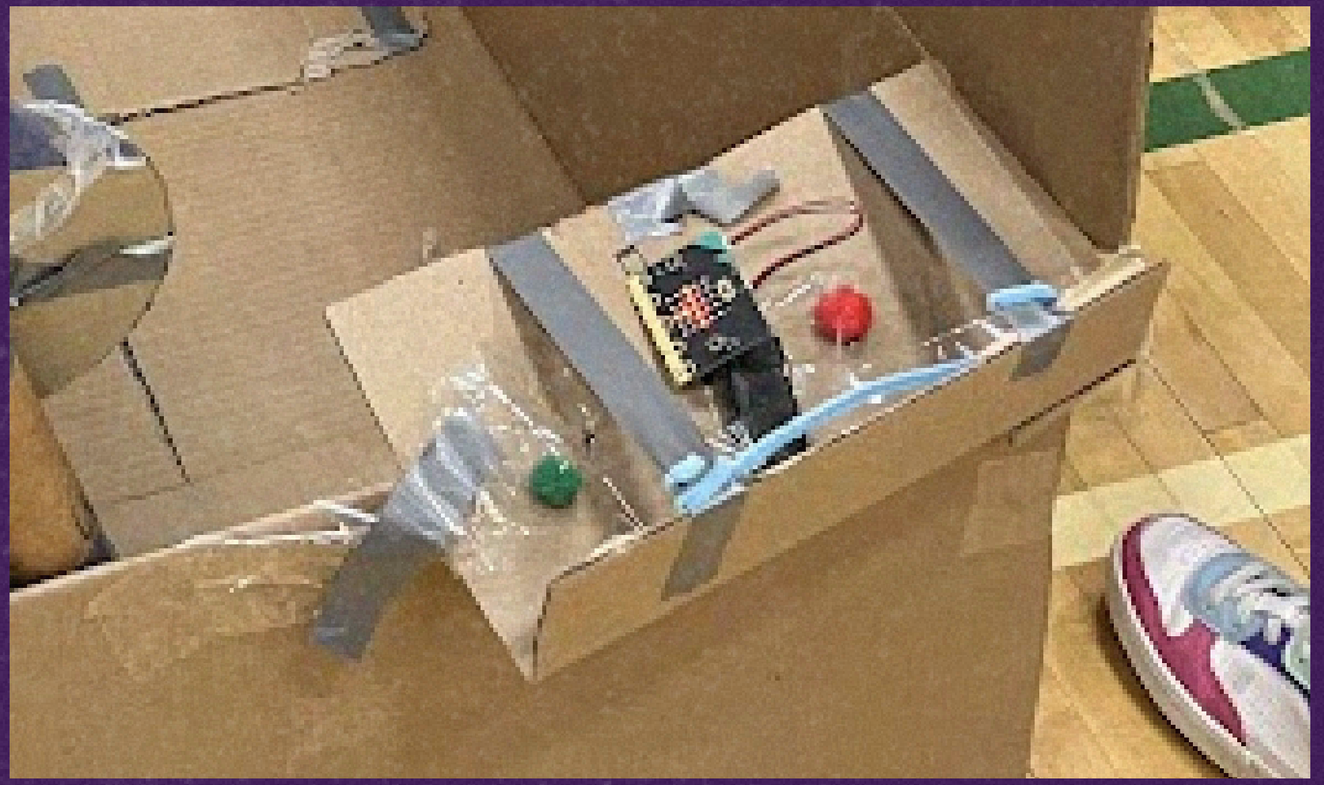
Creativity in Design

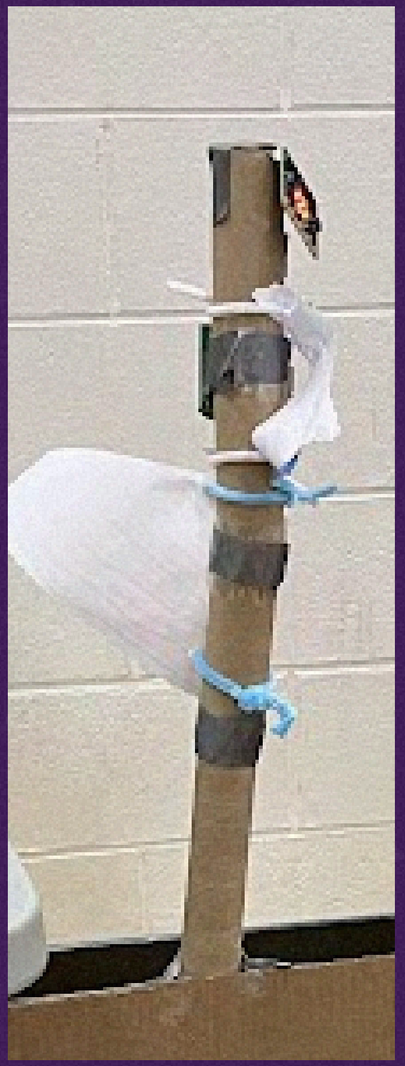
Fallingwater by Frank Lloyd Wright

Smartphone Interfaces

Chicago's "Cloud Gate" (The Bean)

Creativity in Design





Every single K-3 wanted to ride in the boat!

Design Thinking in Action

Defining

Understanding the problem that needs to be solved or the need that will be filled

Figuring out the goal is the first step to design thinking. This step often includes using empathy.

Ideation Phase

Encouraging brainstorming and collaboration among students

This phase promotes diverse ideas, fostering teamwork and innovation in classrooms.

Prototyping

Creating tangible solutions to test concepts and ideas

Prototyping allows students to visualize their solutions and refine them effectively.

Testing and Feedback

Gathering insights to improve designs and approaches

Feedback helps students iterate, enhancing their projects through constructive criticism.

Reflection

Evaluating the process and celebrating learning outcomes

Reflection enables students to understand their growth and apply lessons learned.

Real World Design Thinking

1. Healthcare: Improving Patient Experience
   - Cleveland Clinic used design thinking workshops with doctors, nurses, and patients to reimagine hospital experiences.
   - They redesigned waiting rooms, improved doctor–patient communication systems, and created “empathy training” for staff.
   - Result: Higher patient satisfaction and stronger trust.

2. Education: Reimagining Classrooms
   - Stanford’s d.school applied design thinking to K–12 learning. Teachers and students co-created classroom layouts and lesson delivery styles.
   - Examples: flexible furniture for collaboration, and project-based learning structures.
   - Result: More engaged students and adaptable learning spaces.

3. Technology: Apple’s Product Development
   - Apple consistently uses design thinking principles—empathy for user needs, simple and intuitive design, and iterative prototyping.
   - The iPhone emerged from understanding people’s desire for one device combining phone, music, and internet browsing.
   - Result: A product that reshaped global communication and lifestyle.

4. Public Services: Government Redesign
   - The UK Government Digital Service (GDS) applied design thinking to improve online government services.
   - They focused on user-centered design: simpler forms, clearer instructions, mobile-friendly platforms.
   - Result: Millions saved and much easier access for citizens.

Real World Design Thinking

5. Social Innovation: Clean Water Access
   - IDEO.org partnered with communities in developing countries to design low-cost clean water solutions.
   - Through empathy interviews, they learned about cultural habits around water use, leading to context-appropriate filtration systems.
   - Result: Broader adoption and sustained use of safe water technology.

6. Retail: Shopping Experience
   - Nike used design thinking to reinvent their retail stores as interactive experiences.
   - Stores now feature areas for product testing, personalization stations, and immersive digital displays.
   - Result: Increased customer loyalty and stronger brand engagement.

7. Transportation: Urban Mobility
   - Uber and Lyft both started from understanding pain points in traditional taxi services—difficulty hailing, uncertainty in fares, and payment hassles.
   - By prototyping a simple app and iterating with user feedback, they transformed urban mobility worldwide.

8. Nonprofits: Tackling Hunger
   - The Food for All project in Boston used design thinking to address food waste and hunger.
   - They connected restaurants with surplus meals to people in need through a simple app.
   - Result: Reduced food waste while feeding thousands.

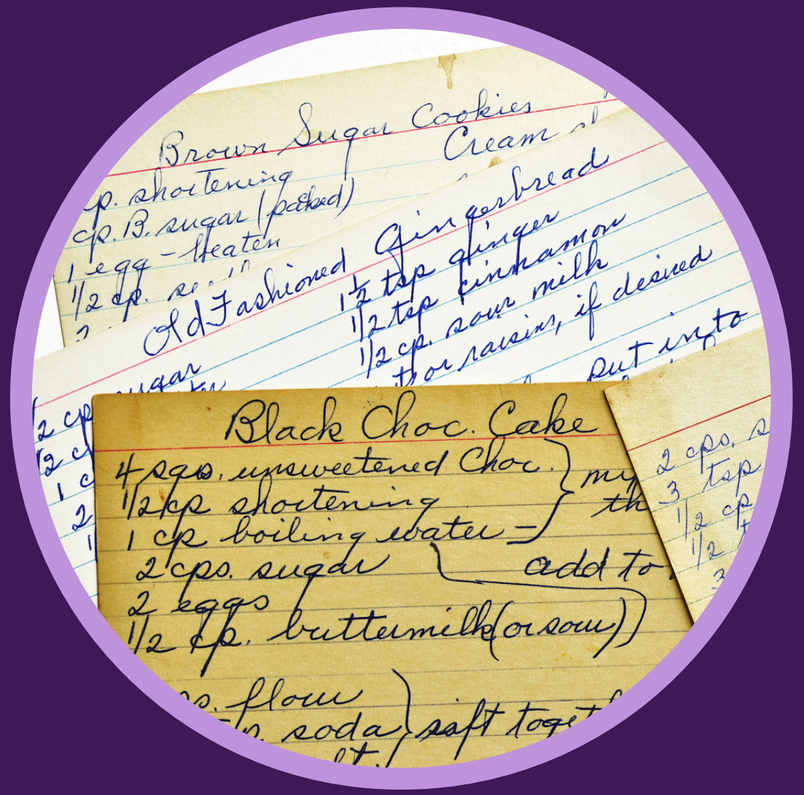
Computational Thinking & Coding

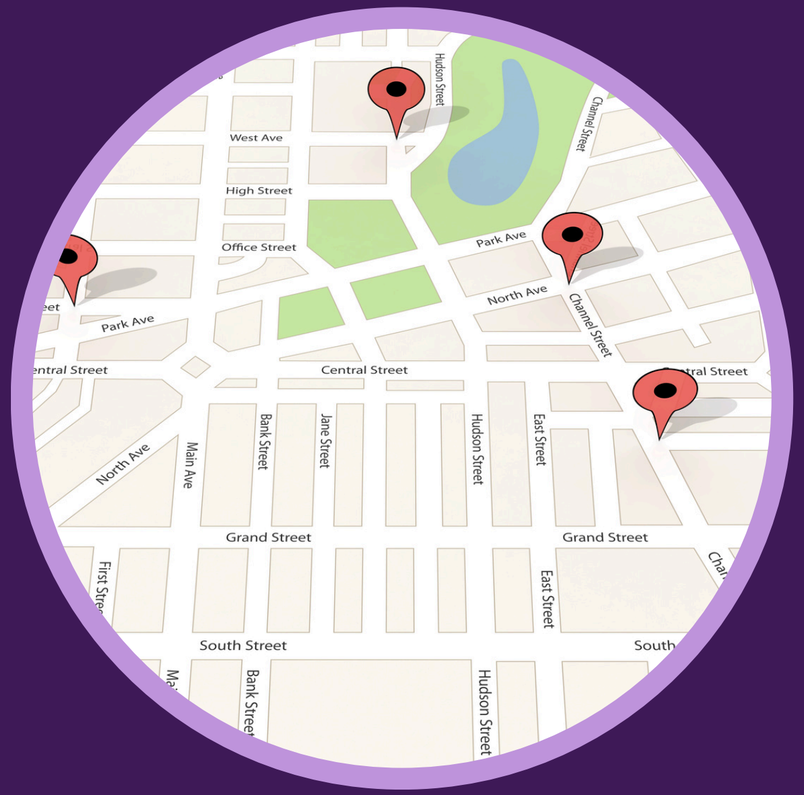
What do you need to know to teach your students to code?

Algorithm
Debug
Input
Output

Algorithm

In computer science, an algorithm is a step-by-step set of instructions designed to solve a problem or accomplish a specific task. Algorithms are the foundation of programming and are used to structure the logic behind software, apps, and systems. They must be clear, finite, and effective—meaning they eventually produce an output after a certain number of steps.

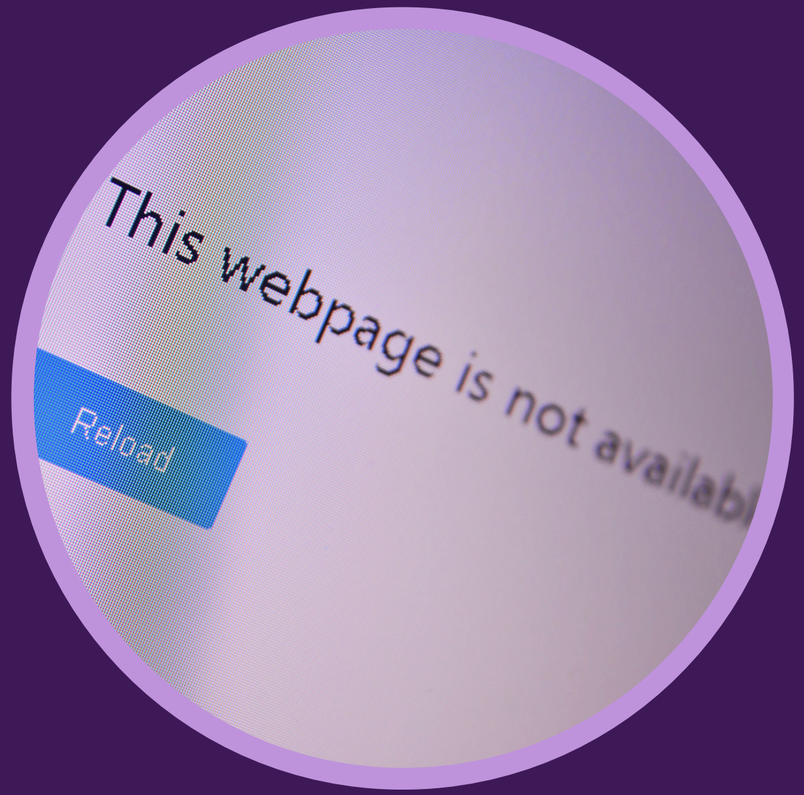




Debug

In computer science, to debug means to find and fix errors (bugs) in code or a system so that it works as intended.

Debugging is the problem-solving process of detecting, isolating, and correcting those mistakes.



Input & Output

Input = The data, signals, or instructions given to a computer system to be processed.
Output = The result or response produced by the computer system after processing the input.

Think of it as: Input → Processing → Output

Block Based Programming

Scratch
Makecode (micro:bit)
block.ly Games
code.org
Tynker

Integration Strategies for CS

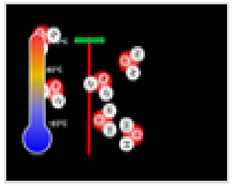
Practical ways to incorporate computer science into everyday lessons

Integrating computer science into your curriculum can enhance student engagement. Teachers can foster a deeper understanding of computational concepts with and without technology. These hands-on techniques promote critical thinking and collaboration among students.

playngel

[Mrs. D's Scratchtivities]

[etwithmrsd.com]



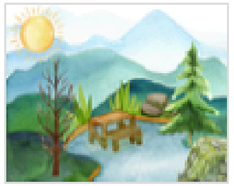
Temperature Model Example

Last modified: 12 Jun 2023

See inside

3 0 0 0 0 0

Unshare



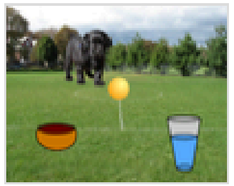
Water Cycle Example

Last modified: 4 Jun 2023

See inside

2 0 0 0 0 0

Unshare



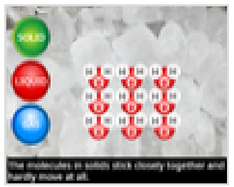
Virtual Pet Example

Last modified: 4 May 2023

See inside

2 0 0 0 0 0

Unshare



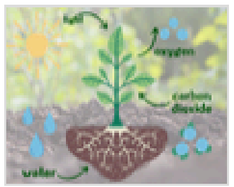
State of Water Example

Last modified: 12 Jun 2023

See inside

8 1 0 0 0 0

Unshare



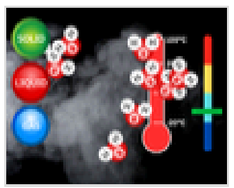
Photosynthesis Example

Last modified: 13 Jun 2023

See inside

4 0 0 0 0 0

Unshare



Particle Model Example

Last modified: 18 Jun 2023

See inside

8 0 0 0 0 0

Unshare

Gr. 4 Matter	Gr. 4 Energy	Gr. 4 Earth Systems
• **Simulation & Modeling**: Use coding (e.g., Scratch, PhET-style simulations) to model recycling processes or waste management cycles. Students can build flowcharts or simple animations showing reduce → reuse → recycle. • **Data Tracking:** Have students create a spreadsheet or coded app to track their personal or classroom waste reduction plans and visualize progress with graphs. • **Hazard Symbol Recognition**: Build a digital quiz where students classify hazard symbols using if-then logic, reinforcing both science safety and CS algorithms.	• **Algorithms for Forces**: Students can write step-by-step instructions (algorithms) predicting what will happen when magnets or gravity act on objects. • **Simulations**: Create simple coded experiments that simulate how magnets attract/repel or how gravity changes with distance. • **Digital Data Collection**: Use sensors (e.g., Micro:bit accelerometers) to measure forces acting on objects and log results digitally.	• **Systems Modeling**: Students code or diagram models showing interactions among the lithosphere, atmosphere, hydrosphere, and biosphere (e.g., water cycle animations). • **Data Visualization**: Collect local weather or conservation data, then represent it with charts using digital tools. • **Scenario Simulations**: Program simple “what if” scenarios: what happens to ecosystems if water becomes polluted or temperature rises?

Gr. 4 Living Systems	Gr. 4 Space	Gr. 4 Scientific Method
• **Classification Algorithms**: Students design a decision-tree algorithm (on paper or in Scratch) to classify plants and animals by traits like appearance or habitat. • **Digital Observation Logs:** Create coded forms or spreadsheets to record data on local organisms' external structures or sensory responses. • **Virtual Investigations**: Simulate plant responses to light or touch with animations that "grow" toward light or shrink away from obstacles.	• **Constellation Mapping**: Students use coding to create interactive star maps where clicking on a star reveals its name and cultural story. • **Orbital Simulations**: Program animations that show Earth's rotation and revolution, explaining day/night and seasons. • **Calendars & Timekeeping**: Build digital timelines or simple apps comparing lunar vs. solar calendars, reinforcing cultural and scientific perspectives.	• **Data Collection & Representation**: Students gather qualitative/quantitative data and use digital tools (coding graphs, spreadsheets, or simulations) to present results. • **Reliability Testing**: Program simulations that test the same experiment multiple times to see consistent or inconsistent outcomes. • **Evidence & Conclusions**: Use coding to design simple models where changing inputs (variables) shows how evidence supports or challenges a conclusion.

Gr. 5 Matter

- **Particle Simulations:** Use block coding (e.g., Scratch) to animate particle movement in solids, liquids, and gases. Students can adjust variables (spacing, movement speed) to represent states of matter.
- **Density Calculators:** Create simple coded calculators or spreadsheets that compare mass and volume to determine if an object will sink or float.
- **Data Collection Tools:** Program digital logs where students input measured mass/volume and generate density graphs.

Gr. 5 Energy

- **Forces in Air & Water Simulations:** Use coding to model thrust, drag, lift, and weight on a flying object. Students can change variables like wing size or thrust power to see different outcomes.
- **Virtual Buoyancy Tests:** Students can code simulations testing whether objects float or sink based on density vs. fluid properties.
- **Renewable vs. Non-renewable Comparison:** Build interactive digital posters or apps where clicking icons (e.g., wind, fossil fuels) reveals advantages, disadvantages, and impacts.

Gr. 5 Earth Systems

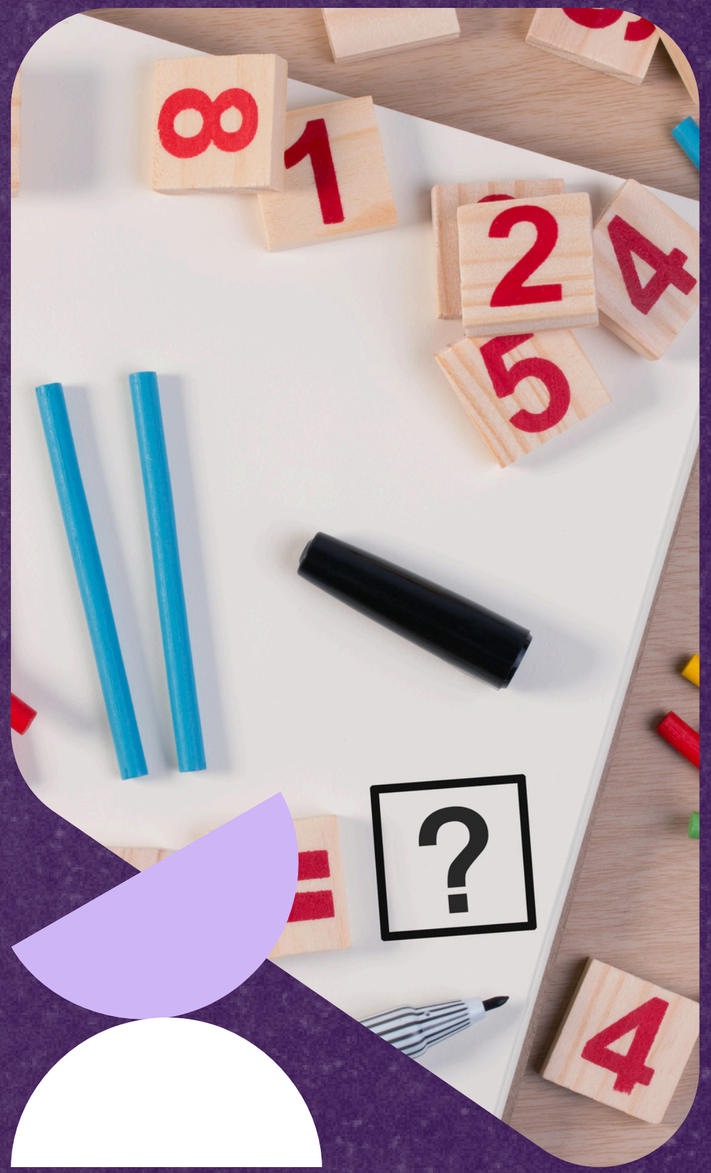
- **Weather Data Dashboards:** Use spreadsheets or simple coding to track and graph local weather data (temperature, precipitation, wind). Compare with climate averages.
- **Climate Zone Classifier:** Students design an algorithm (decision tree) to classify regions as tropical, temperate, etc., based on climate data.

Weather Prediction Models: Explore how computer models simulate climate by letting students “train” a simple simulation using historical weather data.

Gr. 5 Living Systems	Gr. 5 Space	Gr. 5 Scientific Method
• **Body System Models:** Code interactive diagrams where clicking on digestive, respiratory, or circulatory systems shows their functions. • **Simulation of Processes:** Use coding to model how oxygen and nutrients move through systems (respiratory → circulatory → musculoskeletal). • **Plant Transport Simulation:** Animate how xylem and phloem carry water/nutrients/sugars, mirroring how data flows in a program.	• **Astronomical Phenomena Animations:** Code moon phases, eclipses, or Earth's tilt to show seasons. Add loops to simulate cycles. • **Digital Calendars:** Students build simple digital lunar or solar calendars, showing how astronomical cycles link to daily life and agriculture. • **Cultural Stories & Tech Integration:** Use digital storytelling tools to represent Indigenous legends connected to constellations or auroras, paired with coded visualizations.	• **Controlled Experiment Simulators:** Students design a program where changing one variable (e.g., temperature, light) affects an outcome (e.g., plant growth). • **Bias & Data Representation:** Explore how data can be misrepresented by creating different graphs of the same dataset and comparing accuracy. • **Ethics in Tech:** Discuss how coding and digital data collection require ethical handling—no altering results, ensuring privacy, and minimizing harm.

Gr. 6 Matter	Gr. 6 Energy	Gr. 6 Earth Systems
• **Chemical Reaction Simulations:** Students can use coding to model how substances change when combined, showing inputs (reactants) and outputs (products). • **Interactive Periodic Table:** Build a digital tool where selecting an element shows its properties, uses, and safety notes. • **Waste Reduction Data Tracking:** Program spreadsheets or apps to track recycling rates, composting, and landfill diversion.	• **Energy Conversion Simulations:** Code animations showing energy transfer (e.g., electrical → heat in a toaster). Students can change variables to see different efficiencies. • **Circuit Design with Microcontrollers:** Use Micro:bit or Arduino to design coded circuits, testing how switches, resistors, or sensors affect flow. • **Data Logging:** Build digital logs to record electricity use in classrooms and visualize conservation strategies.	• **Dynamic Planet Models:** Simulate the rock cycle with animations that loop through erosion, heat, and pressure. • **Natural Disaster Modeling:** Students code simple earthquake, volcano, or landslide simulations showing cause–effect relationships. • **Climate Impact Dashboards:** Use spreadsheets or apps to visualize human impacts on Earth systems, such as CO_2 levels and their consequences.

Gr. 6 Living Systems	Gr. 6 Space	Gr. 6 Scientific Method
• **Human Body Coding Models:** Build interactive diagrams of organs or systems (e.g., nervous, circulatory) showing how signals and energy flow, similar to computer networks. • **Adaptation Simulations:** Students design “virtual organisms” with traits coded to survive in specific ecosystems, then test survival in different conditions. • **Biodiversity Data Collection:** Program digital biodiversity trackers that classify and log observed organisms with photos or traits.	• **Space Technology Models:** Code animations of satellites orbiting Earth, showing communication or GPS functions. • **Exploration Simulators:** Students program “Mars rover” robots (virtual or physical) to navigate and collect data. • **Astronomical Data Tools:** Use coding to graph real NASA datasets, such as solar activity or planetary distances.	• **Experiment Planning Algorithms:** Students design flowcharts or pseudocode to outline controlled investigations step by step. • **Digital Experiment Logs:** Use spreadsheets or apps for recording trials, ensuring reliable and replicable data. • **Data Integrity Lessons:** Explore how changing variables in a simulation must be transparent and ethical, reinforcing reliability of conclusions.



Mathematics

- Algorithms for Math Operations: Students write step-by-step pseudocode or block code for multiplication, division, or fractions.
- Pattern Recognition: Use coding to generate geometric patterns, tessellations, or fractals.
- Data & Graphing: Collect class survey data, then use spreadsheets or coded visualizations to represent averages, ratios, and probability.
- Coding Math Games: Students design Scratch games where players solve math problems to progress.

English Language Arts & Literature

- Storytelling with Code: Use Scratch to create interactive choose-your-own-adventure stories (branching algorithms = plot choices).
- Text Analysis: Introduce word frequency counters using simple scripts or spreadsheets to analyze texts/poems.
- Sequencing: Compare story structure (beginning → middle → end) to algorithms (input → process → output).
- Digital Publishing: Students code or design e-books with clickable elements, hyperlinks, or animations.

Social Studies

- Simulating Communities: Program simple simulations of trade, resource use, or government decision-making.
- Timelines & History Visualizations: Create digital timelines with interactive coding (scroll, click for details).
- Mapping & Data Layers: Use geographic information systems (GIS-lite tools) or coding maps to show population, migration, or environmental change.
- Decision Trees: Explore cause-and-effect in historical events with branching logic (e.g., “What if Canada chose X instead of Y?”).

Art

- Generative Art: Use coding platforms like Turtle Art or Processing to create algorithmic art pieces.
- Symmetry & Geometry in Code: Students code designs that mirror, rotate, or repeat shapes.
- Animation: Create short animations of stories or abstract concepts with Scratch, emphasizing both art and computational design.
- Pixel Art: Introduce digital art grids to show how computers represent images in binary.

Music

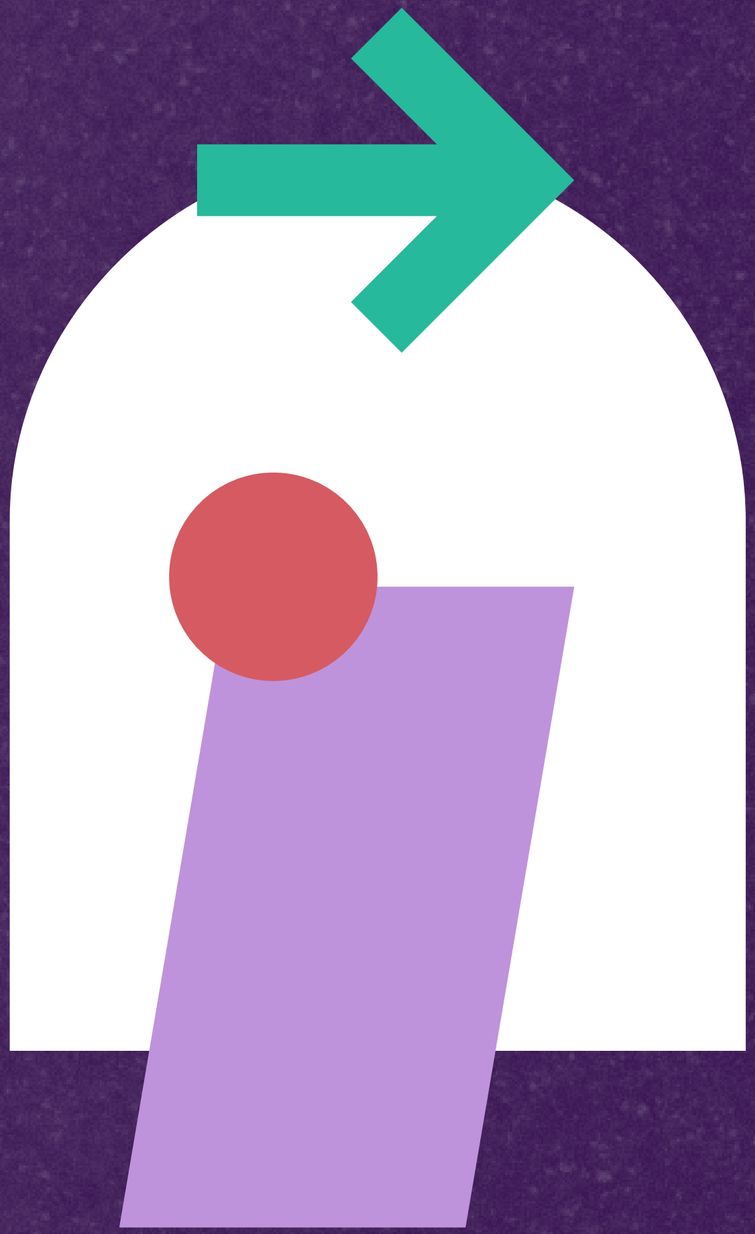
- Coding Music: Use Sonic Pi or Scratch extensions to code beats, loops, and melodies.
- Rhythm Patterns as Algorithms: Break music into repeatable instructions (loops and conditionals).
- Digital Composition: Create visual sound maps where coding triggers instruments.
- Data → Music: Convert data sets (weather, class survey responses) into rhythms or tones.

Health & Wellness

- Data Tracking: Code or use spreadsheets to log food groups, sleep, or physical activity, then visualize results.
- Simulations: Model body systems (e.g., how exercise increases heart rate and oxygen use).
- Decision-Making Algorithms: Students design flowcharts for healthy choices (e.g., "If I feel tired → then choose rest or exercise?").
- Digital Safety: Teach algorithmic thinking through online safety scenarios (what-if decision trees).

Physical Education

- Wearable Tech Integration: Use pedometers/heart-rate monitors to gather exercise data and visualize in graphs.
- Algorithmic Games: Students design playground games using coding principles (loops = repeated actions, conditionals = if a ball is caught, then...).
- Movement Sequences as Code: Have students “program” each other with step-by-step dance or sport moves.
- Performance Analysis: Use slow-motion video and digital tagging to analyze form and provide feedback.



angela.dearing@wrps11.ca

APLC Site & YouTube

Website

etwithmrsd.com

Survey

https://aplc.ca/survey/?id=15077