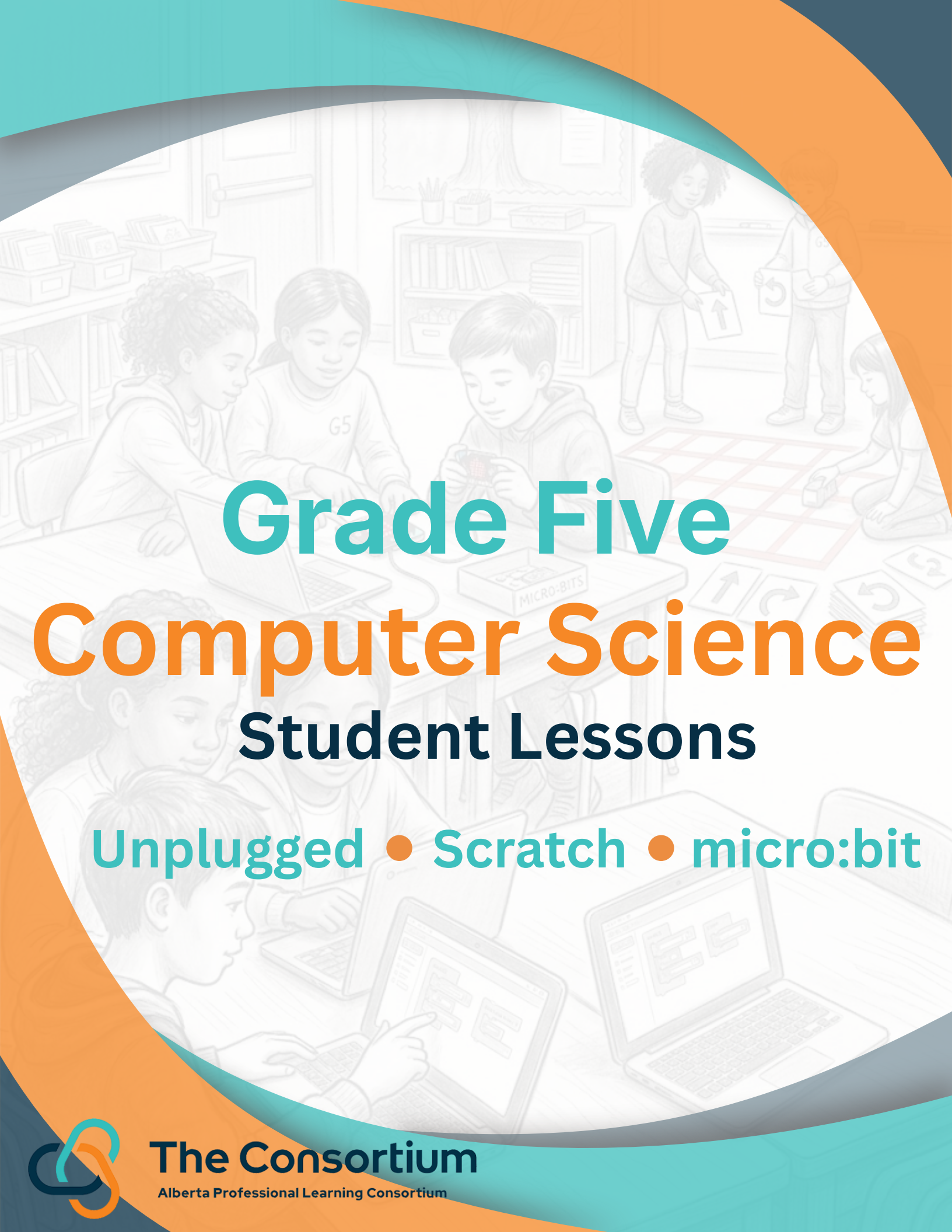

Grade Five Computer Science

Student Lessons

Unplugged • Scratch • micro:bit

The Consortium
Alberta Professional Learning Consortium

Grade 5 CS Chapter 1 Lesson Plans

Lesson 1.1: The "Analog to Digital" Blueprint (Unplugged)

Big Idea: Before we build on a computer, we use computational thinking to design a plan—which is an artifact itself!

🎯 Learning Objectives

Students will define a **computational artifact** as anything created by a human using a computer or a plan for a machine to execute.

Students will design a **paper prototype** (an artifact) to solve a specific classroom need.

Students will evaluate their design based on **functionality** and **usability**.

⏱️ Lesson Outline (60 Minutes)

The Hook: What counts? (10 Mins): Show the class a printed photo, a hand-drawn map, and a digital tablet. Ask: "Which of these is a computational artifact?" Reveal that the tablet content and even the *plan* for it are artifacts.

The Mission: The App Architect (10 Mins): Students must design a "Homework Helper" app interface on paper. They must interview a partner to see what they struggle with most (e.g., forgetting due dates, messy notes).

Hands-On: Paper Prototyping (30 Mins):

Sketch (10 mins): Draw the main screen. Where do the buttons go?.

Iterate (20 mins): Trade with a partner. Can they "use" the paper app easily? Use their feedback to refine the design.

Reflection (10 Mins): Discuss how the paper plan is a "computational artifact" because it uses computational thinking to solve a problem.

Lesson 1.2: The "Emotion Machine" (micro:bit)

Big Idea: Code is a language used to create artifacts that express ideas or feelings.

🎯 Learning Objectives

Students will relate a **block of code** to a specific behavior on the micro:bit.

Students will create a computational artifact (an emoji display) that meets a user's **aesthetic** and **functional** needs.

⏱️ Lesson Outline (60 Minutes)

The Hook: Binary Feelings (10 Mins): Discuss how **images** and **graphics** are computational artifacts. Can a small 5×5 LED grid be an image?

The Mission: The Desk Mood-Ring (10 Mins): Students design a program that displays an icon representing their current "working status" (e.g., "I'm focused," "I need help," or "I'm finished").

Hands-On: Coding the Artifact (30 Mins):

Students use a **visual block-based language** (MakeCode) to show different icons when Button A or B is pressed.

Loop it: Add a repeating animation (a loop) to show the artifact is "active".

Gallery Walk (10 Mins): Students walk around the room to see how different "Desk Mood-Rings" look and function.

Lesson 1.3: The Interactive Storyboard (Scratch)

Big Idea: Multimedia artifacts (video, audio, and images) can be combined using code to share ideas.

🎯 Learning Objectives

Students will combine multiple types of artifacts (**audio, images, and code**) into one project.

Students will explain what happens when **multiple blocks of code** are executed.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Digital Mashup (10 Mins): Look at a website. Identify the different artifacts: text, images, and the code holding them together.

The Mission: The "Welcome to Grade 5" Guide (10 Mins): Create an interactive Scratch project that introduces a new student to the classroom.

Hands-On: Building the Artifact (30 Mins):

Images: Choose or draw a backdrop of the school.

Audio: Record a "Welcome" message (an audio artifact).

Code: Use blocks to make a character move when clicked and play the recording.

Peer Review (10 Mins): Students test each other's projects and propose one **enhancement** to improve **usability**.

Gr 5 CS Chapter 2 Lesson Plans

Lesson 2.1: The Human Algorithm (Unplugged)

Big Idea: Before we write code, we must design a clear, step-by-step "recipe" (algorithm) that a machine can follow.

🎯 Learning Objectives

Students will define an **algorithm** as a step-by-step set of instructions to solve a problem.

Students will identify the **functionality** (purpose) of a set of instructions.

Students will practice "debugging" an algorithm to improve its **usability**.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Robot Chef (10 Minutes):

The teacher acts as a "Robot Chef" who only understands literal instructions.
The Goal: Make a sandwich.

The Chaos: If a student says "Put the jam on the bread," the teacher puts the *whole jar* on the loaf.
The Lesson: Computers need every tiny step planned out in a specific order.

The Mission: The Obstacle Course Blueprint (10 Minutes):

Students must design an algorithm to guide a blindfolded "Robot" (a partner) through a simple classroom obstacle course (around a desk, under a chair).

Hands-On: Drafting the Recipe (30 Minutes):

Interview (5 mins): Partners discuss potential safety hazards in the course.

Sketch & Write (10 mins): In their **Designer Notebook**, students write the steps (e.g., "Walk 3 steps," "Turn 90 degrees left").

Test & Iterate (15 mins): One student acts as the "Computer" and follows the code exactly. If the "Robot" hits a desk, the "Programmer" must fix the instructions.

Reflection (10 Minutes):

Discuss: "Was your algorithm **reliable**? Did it work the same way every time?".

Lesson 2.2: The Digital Light Show (micro:bit)

Big Idea: Code is the "Robot Language" we use to translate our algorithms into actions.

🎯 Learning Objectives

Students will **translate** a written algorithm into block-based code.

Students will prioritize **aesthetics** by designing a pleasing visual sequence on the LED grid.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Secret Signal (10 Minutes):
Discuss how light signals (like lighthouses or traffic lights) are functional artifacts.

The Mission: The "SOS" Emergency Flasher (10 Minutes):
The goal is to create a flashing light pattern that is **reliable** (works every time) and **efficient** (not too many unnecessary blocks).

Hands-On: Coding the Algorithm (30 Minutes):
The Algorithm: 1. Turn on all LEDs. 2. Wait 1 second. 3. Turn off all LEDs. 4. Wait 1 second. 5. Repeat.
The Translation: Students use the forever loop and show leds blocks in MakeCode to match the algorithm.

Aesthetic Challenge: Can you make the light "pulse" or "fade" by changing the icons?.

Gallery Walk (10 Minutes):
Compare different "flasher" designs. Which one has the best **usability** (is the easiest to understand as a signal)?.

Lesson 2.3: The Efficient Assistant (Scratch)

Big Idea: Great digital artifacts use **efficiency** and **functionality** to solve problems quickly.

🎯 Learning Objectives

Students will build an interactive artifact that performs a specific **function**.

Students will evaluate the **efficiency** of their code.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Fast-Food App (10 Minutes):

Look at a delivery app. Why is it **efficient**? (It finds the shortest route) .

The Mission: The Instant Translator (10 Minutes):

Students will design a program where a character (Sprite) translates a "click" into a specific sound or action instantly.

Hands-On: Building the Assistant (30 Minutes):

Functionality: Use When Sprite Clicked to make the character play a "Boing" sound or change color.

Usability: Add a "Start" button so the user knows exactly how to begin.

Efficiency: Challenge students to use the fewest blocks possible to achieve the goal.

Reflection & Debug (10 Minutes):

Students swap seats. If they can't figure out how to start the program in 5 seconds, the **usability** needs work!.

Gr 5 CS Chapter 3 Lesson Plans

Lesson 3.1: The Spiral Staircase (Unplugged)

Big Idea: Designers don't get it right the first time; they use **iteration** to make their ideas "climb higher."

🎯 Learning Objectives

Students will define **iteration** as a process of repeating a task with improvements.
Students will distinguish between **enhancing** (adding value/features) and **refining** (polishing/adjusting).
Students will develop multiple iterations of a physical artifact.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Perfect Paper Plane (10 Mins):
Give students 2 minutes to build a paper plane and fly it.
Ask: "Did it hit the target? No? That's Prototype 1."
Explain that iteration is like a spiral staircase—you go in a circle, but you move higher each time.

The Mission: The "Ultimate Bookmark" (10 Mins):
Students must design a bookmark that doesn't just hold a page, but solves a problem (e.g., it falls out, it doesn't show exactly which line you were on).

Hands-On: The Iterative Loop (30 Mins):
Iteration 1 (Prototype): Build a quick version using cardstock.
Enhancing (The Upgrade): Students must add a "value" feature (e.g., a clip so it won't fall out).
Refining (The Polish): Students use feedback to make a small adjustment (e.g., smoothing a sharp edge or making the clip tighter).

Reflection (10 Mins):
Discuss: "How did your final version meet the user's needs better than the first one?"

Lesson 3.2: The Adaptive Icon (micro:bit)

Big Idea: Digital artifacts are refined through testing to improve their **reliability**.

🎯 Learning Objectives

Students will apply the design process to **refine** an existing piece of code.
Students will use feedback to identify where an artifact lacks **functionality**.

⏱️ Lesson Outline (60 Minutes)

The Hook: The "Buggy" Badge (10 Mins):

Show a micro:bit flashing a "Happy Face" so fast you can barely see it. Ask: "Is this functional? Does it meet the need of showing an emotion?"

The Mission: The Name Tag Upgrade (10 Mins):

Students start with a basic scrolling name tag. They must iterate on it twice to make it "professional quality."

Hands-On: Enhancing and Refining (30 Mins):

Version 1: Just the name scrolling.

Iteration 2 (Enhancement): Add a feature—make an icon appear *after* the name when Button A is pressed.

Iteration 3 (Refinement): Adjust the **speed** of the scroll. Is it too fast to read? Too slow? This is refinement!

Peer Review (10 Mins):

Students trade micro:bits. The "User" gives one piece of feedback: "I wish it did [X]" or "The [Y] is too blurry."

Lesson 3.3: The Evolution of a Sprite (Scratch)

Big Idea: In software, we use iterations to ensure **usability** and **aesthetics** are balanced.

🎯 Learning Objectives

Students will develop multiple versions of a Scratch project.

Students will collaborate with others to propose **enhancements** to a digital artifact.

⏱️ Lesson Outline (60 Minutes)

The Hook: Game Updates (10 Mins):

Talk about popular games (like Minecraft or Roblox). Do they look the same as they did 5 years ago? No! They have been "iterated" thousands of times.

The Mission: The Catching Game (10 Mins):

Students will build a simple game where a sprite follows the mouse to catch a falling object.

Hands-On: Rapid Iteration (30 Mins):

Prototype 1: Get the sprite moving.

Iteration 2 (Enhancement): Add a score counter (adding functionality).

Iteration 3 (Refinement): Change the colors or sounds to make it more pleasing (aesthetics).

Collaboration (10 Mins):

Students "Playtest" a neighbor's game. They must write down one **Refinement** (a small fix) and one **Enhancement** (a big idea) for their partner's next iteration.

Gr 5 CS Chapter 4 Lesson Plans

Lesson 4.1: The Design Firm (Unplugged)

Big Idea: Collaboration is a "superpower" that brings different skills, experiences, and perspectives together to solve a problem.

🎯 Learning Objectives

Students will identify the benefits of **collaboration** (sharing workload, brainstorming, and finding errors).
Students will contribute a specific skill (drawing, planning, or speaking) to a group project.
Students will propose enhancements to a teammate's design.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Teamwork Challenge (10 Mins):
Try to tie your shoes using only one hand. Now, have a partner use one of *their* hands to help you.
Discuss: Why was it easier with two people? (Different perspectives and shared workload).

The Mission: The Inclusive Playground (10 Mins):
Groups of three are "Design Firms." Their task is to design a playground that is safe and fun for everyone (including students with different physical abilities).

Hands-On: Collaborative Blueprinting (30 Mins):
Role Assignment: One student is the **Safety Officer**, one is the **Architect** (aesthetics), and one is the **User Expert** (usability).
Brainstorm: Each student draws one piece of equipment.
The Merge: They must combine their ideas into one master blueprint. If the Architect draws a tall slide, the Safety Officer must propose a "refinement" to make it safer.

Reflection (10 Mins):
Discuss: "What is one thing your partner noticed about your design that you missed?".

Lesson 4.2: The Shared Code-Base (micro:bit)

Big Idea: Digital artifacts are stronger when "fresh eyes" look at the code to spot bugs or confusing features.

🎯 Learning Objectives

Students will collaborate to **debug** a partner's code.
Students will use feedback to improve the **usability** of a micro:bit interface.

⏱️ Lesson Outline (60 Minutes)

The Hook: The "Mystery" Device (10 Mins):
Hand a student a micro:bit with a pre-loaded program that does something cool but has no instructions.
Ask: "Is it easy to use? What's missing?" (A collaborator would have suggested a 'Start' message!).

The Mission: The Multi-Tool (10 Mins):
Pairs will build a "Hiker's Tool" that has a compass (Button A) and a step counter (Button B).

Hands-On: Pair Programming (30 Mins):
Partner A (The Driver): Controls the mouse and keyboard.
Partner B (The Navigator): Watches for errors and suggests "enhancements" (e.g., "Let's add a sound when a step is counted!").
Switch: Every 10 minutes, students swap roles.

Peer Review (10 Mins):
Trade devices with another pair. Each pair must find one "usability" issue (e.g., "The icons are too small") and suggest a fix.

Lesson 4.3: The Remix Project (Scratch)

Big Idea: Sharing the workload allows designers to focus on their strengths to create complex multimedia artifacts.

🎯 Learning Objectives

Students will explain how **multiple perspectives** improve a digital design.
Students will contribute to a shared digital artifact by focusing on a specific element (audio, code, or art).

⏱️ Lesson Outline (60 Minutes)

The Hook: The Movie Credits (10 Mins):
Watch the end of a short animated clip. Look at how many people worked on it (Animators, Sound Designers, Writers). Why didn't one person do it all?

The Mission: The Classroom "How-To" Guide (10 Mins):
In groups of three, students create a Scratch animation teaching a skill (e.g., "How to use the library").

Hands-On: The Creative Team (30 Mins):
Student 1 (The Artist): Designs the sprites and backdrops (Aesthetics).
Student 2 (The Sound Engineer): Records instructions and adds music (Audio Artifacts).
Student 3 (The Coder): Snaps the blocks together to make everything move (Functionality).

The Integration (10 Mins):
The team brings all three parts together. They must "Iterate" if the sound doesn't match the animation.

Gr 5 CS Chapter 5 Lesson Plans

Lesson 5.1: The Human Translator (Unplugged)

Big Idea: Code is simply a language used to translate human plans (algorithms) into a format a machine can execute.

🎯 Learning Objectives

Students will define **code** as a language that can be understood by and run on a computer.
Students will **translate** a physical algorithm into a "symbol-based" code.
Students will explain that a computer cannot think for itself and relies entirely on code.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Alien Architect (10 Mins):

Imagine an alien landed and wanted to build a Lego tower but didn't speak English. Could you use pictures or symbols to tell them what to do?
Explain: "Computers are like that alien. They are powerful but need a specific language—**code**—to function."

The Mission: The Dance Coder (10 Mins):

Students create a 3-step dance "algorithm" (e.g., Clap, Jump, Spin).
They must then "translate" these steps into a symbol code (e.g., 👏, ⬆️, 🔄).

Hands-On: The Translation Station (30 Mins):

Step 1: Write a simple algorithm for a daily task (e.g., getting a drink of water).
Step 2: Create a "Key" where symbols represent actions (e.g., 🖐 = Pick up cup).
Step 3: Swap codes with a partner. The partner must follow the symbols *exactly* as written to see if the "program" works.

Reflection (10 Mins):

Discuss: "If your partner made a mistake, was it because they couldn't think, or because the code was missing a step?" (Reinforce that computers only follow instructions).

Lesson 5.2: The Digital Puzzle (micro:bit)

Big Idea: Visual block-based languages use "puzzle pieces" to make the translation from idea to action easier.

🎯 Learning Objectives

Students will relate a **block of code** to a specific behavior on the micro:bit.
Students will use **drag-and-drop blocks** to design a simple program.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Visual Language (10 Mins):

Show a page of text-based code (Python or C++) vs. a Scratch/MakeCode block. Ask: "Which one looks easier to understand?"

Explain that **Visual Block-Based Languages** are like digital puzzle pieces.

The Mission: The Step-by-Step Counter (10 Mins):

The goal is to make the micro:bit count from 1 to 3 when a button is pressed.

Hands-On: Snapping the Code (30 Mins):

Students open MakeCode.

The Challenge: Find the on button A pressed block. Inside, snap three show number blocks.

Execution: Download the code and run it.

Iteration: If the numbers change too fast to read, what "block" do we need to add? (Introduce the pause block).

Reflection (10 Mins):

Explain what happened when the code was **executed**. Did the computer do exactly what the blocks told it to?

Lesson 5.3: The Sprite's Script (Scratch)

Big Idea: Complex programs are built by combining multiple blocks of code to create a single computational artifact.

🎯 Learning Objectives

Students will explain what happens when **multiple blocks of code** are executed in a sequence.
Students will identify that a computer program is a collection of prepared chunks of instructions.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Script vs. The Actor (10 Mins):
In a play, the actor follows a script. In Scratch, the Sprite is the actor and the **blocks** are the script.

The Mission: The Walking Talker (10 Mins):
Create a program where a Sprite moves across the screen, says a greeting, and changes color.

Hands-On: Sequencing the Script (30 Mins):
Students select a Sprite.
The Script: 1. When Green Flag Clicked, 2. Move 50 steps, 3. Say "Hello Grade 5!", 4. Change Color Effect.
Analyze: Ask students to predict the order. Will the Sprite talk *before* or *after* moving?
Experiment: Change the order of the blocks. How does the **outcome** change?

Collaboration & Peer Review (10 Mins):
Partners look at each other's "scripts." Can they guess what the Sprite will do just by looking at the blocks?

Gr 5 CS Chapter 6 Lesson Plans

Lesson 6.1: The Human Loop (Unplugged)

Big Idea: Loops are a way to make instructions **efficient** by repeating actions without writing them over and over.

🎯 Learning Objectives

Students will define a **loop** as a set of instructions that a computer repeats.
Students will identify the **condition** (the rule) that tells a loop when to stop.
Students will create a physical algorithm that uses a loop to save time and space.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Exhausted Programmer (10 Mins):
The teacher asks students to stand up. Give the instruction: "Clap your hands. Clap your hands. Clap your hands..." repeat this 15 times.
Ask: "Is there a shorter way I could have said that?"
Introduce the **Loop**: "Repeat 15 times: Clap hands."

The Mission: The Fitness Routine (10 Mins):
Students must design a 1-minute workout. Instead of writing every jump jack, they must use a "Repeat" rule.

Hands-On: Coding the Body (30 Mins):
Step 1: Write a "Long Code" (e.g., Step, Step, Step, Step, Hop).
Step 2: Translate it into "Loop Code" (e.g., Repeat 4 times: Step; then Hop).
The Condition Check: Students must include a **Stop Rule** (e.g., "Repeat until the timer hits 30 seconds" or "Repeat 10 times").

Reflection (10 Mins):
Discuss **Efficiency**: "How much paper did you save by using a loop instead of writing every single step?"

Lesson 6.2: The Heartbeat Monitor (micro:bit)

Big Idea: Computers use loops to perform tasks that need to happen constantly or a specific number of times.

🎯 Learning Objectives

Students will translate an algorithm with a loop into **visual block-based code**.
Students will distinguish between a forever loop and a repeat [x] loop.

⏱️ Lesson Outline (60 Minutes)

The Hook: Always On (10 Mins):

Think of a heartbeat or a clock. Does it ever stop and wait for a button? No, it's in a forever loop.

Show how the forever block in MakeCode is a special kind of loop that never meets a "Stop Rule."

The Mission: The Blinking Heart (10 Mins):

Students will create a digital "Heartbeat" artifact that pulses a large heart and a small heart forever.

Hands-On: Snapping the Loop (30 Mins):

Task A: Use the forever block to show a pulse animation.

Task B (The Challenge): Use a repeat 5 times block inside an on button A pressed event. Make the heart "scare" the user by beating fast 5 times and then stopping.

Debug: If the heart doesn't look like it's beating, do you need a pause block inside the loop?

Reflection (10 Mins):

Evaluate the **Reliability**: "Does the loop ensure the heartbeat happens exactly the same way every time?"

Lesson 6.3: The Dance Party (Scratch)

Big Idea: Loops allow us to create complex, rhythmic multimedia artifacts (animations and music) with very little code.

🎯 Learning Objectives

Students will design an algorithm that includes a loop and translate it into Scratch code.

Students will explain what happens when a **loop condition** is met (the loop breaks).

⏱️ Lesson Outline (60 Minutes)

The Hook: The Infinite Dance (10 Mins):

Watch a GIF or a short looping animation. Why doesn't the file size get huge? Because it's just one or two frames repeating in a loop!

The Mission: The Animated Sprite (10 Mins):

Students will choose a Sprite with multiple "costumes" (like a walking person or a flying bat) and make them move across the stage realistically.

Hands-On: Building the Animation Loop (30 Mins):

Functionality: Use the Repeat Until block.

The Condition: Repeat [Next Costume, Move 10 Steps] **until** [Touching Edge].

Aesthetics: Add a "Repeat 10 times" loop for a sound effect (like a drum beat) to accompany the dance.

Peer Review (10 Mins):

Students observe a partner's loop.

The Test: "What is the specific rule (condition) that makes your Sprite stop dancing?"