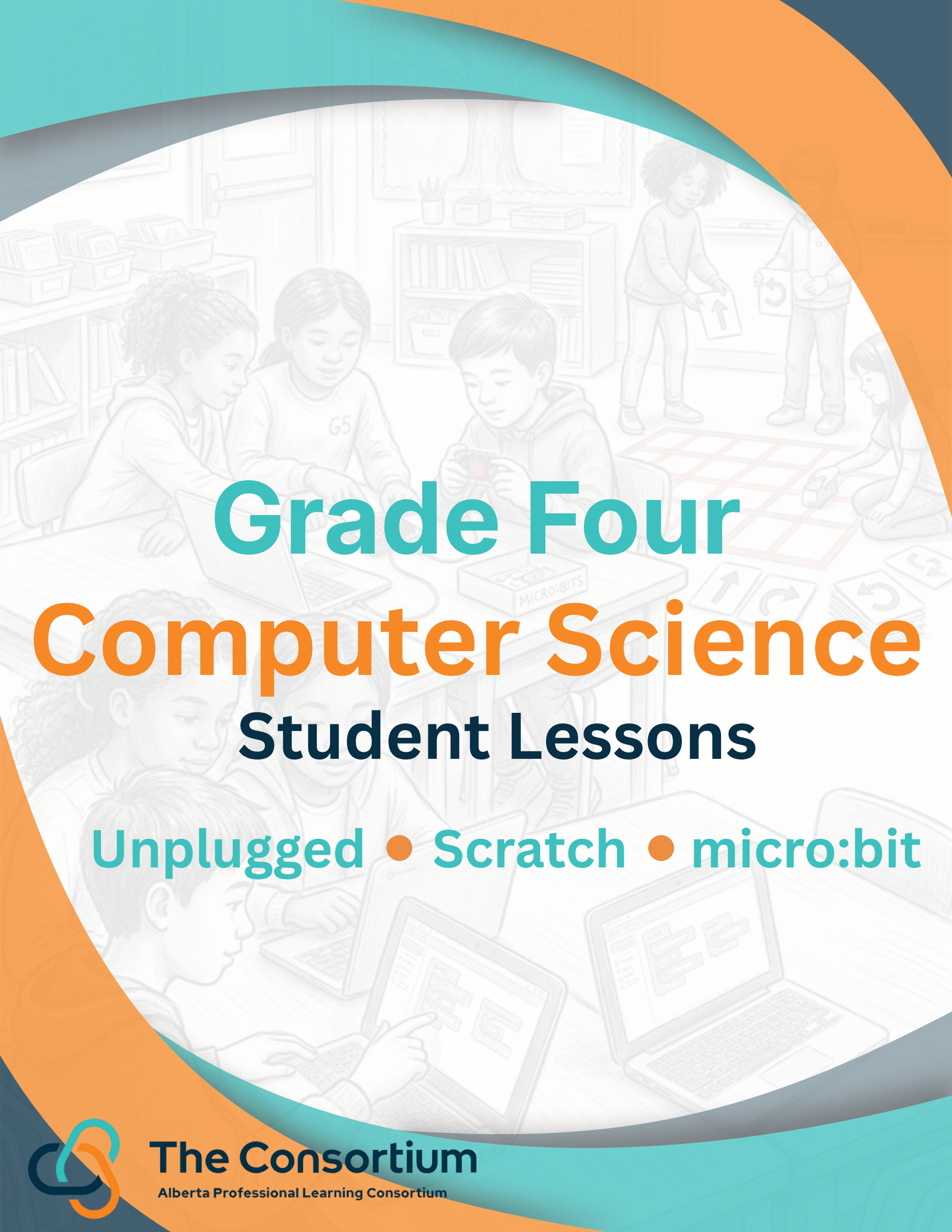

Grade Four Computer Science

Student Lessons

Unplugged • Scratch • micro:bit

The Consortium

Alberta Professional Learning Consortium

Gr 4 CS Introduction Lesson Plan

The "Invisible Tech" Scavenger Hunt

Big Idea: Computer Science isn't just about laptops; it is the invisible force that runs our modern world.

🎯 Learning Objectives

Students will identify "hidden" computers in their daily environment.
Students will understand that humans **design** and **code** technology to solve specific problems.
Students will express the importance of learning CS to become "creators" rather than just "consumers."

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The "Power Outage" Fantasy (10 Minutes)

The Scenario: Tell students to close their eyes. Imagine they woke up today and every single computer in the world stopped working.
The Discussion: Ask, "What would be different?"
Anticipated answers: No YouTube, no iPad games.
The Push: "What about the toaster? The traffic lights? The grocery store checkout? The water treatment plant?"
The Reveal: Explain that we are surrounded by "Invisible Tech." Computer science is the study of how to talk to these machines to make life better.

2. The Mission: The "Hidden Tech" Scavenger Hunt (20 Minutes)

The Activity: Give students a "Scavenger Hunt" checklist. They must walk around the classroom/hallway (or brainstorm their morning routine) to find:
Something that uses a **Sensor** (like a motion-light or automatic sink).
Something that uses a **Screen** to give information.
Something that performs a **Repetitive Task** (like a clock or a bell).
The Connection: For every item found, ask: "What **problem** was the person who designed this trying to solve?"

3. Hands-On: The "Human Remote Control" (20 Minutes)

To show the **WHY** of coding and computational thinking:

The Setup: One student is the "Programmer," and the teacher (or another student) is the "Robot."
The Task: The Robot must put on a jacket or tie a shoe, but it *only* understands very specific, tiny commands (Algorithms).
The Fail: When the Robot puts the jacket on upside down because the "code" wasn't specific enough, it highlights the need for **Troubleshooting** and **Precise Thinking**.
The Lesson: "We learn CS because computers are smart, but they are also very 'obedient.' They need us to be the designers and the thinkers."

4. The "Why" Reflection (10 Minutes)

The Vision: Show a short clip or images of kids using CS to solve big problems (e.g., an app that helps blind people navigate or a robot that cleans the ocean).
Closing Statement: "By the end of this unit, you won't just be someone who *uses* an app. You will be someone who understands how to *build* one. You are moving from being a passenger to being the driver."

🛠️ Materials Needed

Scavenger Hunt Sheets: A simple grid for students to record their "Invisible Tech" finds.
The "Robot" Props: A jacket, a hat, or a backpack for the coding activity.

💡 Connection to the Design Thinking Process

Refer back to the **Student Notes**:

"Today we started with **Understanding the Problem**. We looked at the world and saw where technology is needed. In our next lesson, we will start **Forming Ideas** to solve these problems ourselves!"

Gr 4 CS Chapter 1 micro:bit Lesson Plan

Big Idea: Computer Science uses hardware like the micro:bit to solve human problems through the Design Thinking Process.

🎯 Learning Objectives

Students will apply the **Design Thinking Process** to a digital tool.
Students will understand that sensors (buttons, accelerometers) can gather data from the "User."
Students will create a **functional prototype** using only the built-in features of the micro:bit.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Pocket-Sized Problem Solver (10 Minutes)

The Demo: Show a micro:bit. Ask: "This doesn't look like a robot or a computer yet. How can we make this 'empathize' with a person?"
The Reveal: Explain that the micro:bit has "senses" (buttons, light sensors, motion sensors). Today, we aren't just coding; we are **Troubleshooting** and **Iterating** to make a tool that actually helps someone.

2. The Mission: The "Step-Up" Challenge (10 Minutes)

The Problem: A local gym wants to encourage kids to move more, but kids find standard fitness trackers boring or confusing.
The Goal: Design a **"Micro-Motion Tracker"** or a **"Digital Emotion Picker"** using only the micro:bit.
The Constraint: You cannot use extra wires or LEDs. You must use what is "on-board" (the 5x5 LED grid and the A/B buttons).

3. Hands-On: Digital Prototyping (30 Minutes)

Students work in pairs to move through the non-linear design steps:

Ideate (5 mins): Brainstorm how to use the **A/B Buttons** or **Shake** gesture. (Example: "A" counts a step, "B" shows the total).
Create (15 mins): Use MakeCode to build the script.
Example Code: On Shake -> change count by 1; On Button A -> show number (count).
Analyze & Test (10 mins): Trade micro:bits with another team.
The Test: Does the device "feel" right? Is the LED display clear?
Troubleshooting: If the user thinks the "Shake" is too sensitive, the team must go back to the **Planning** stage to find a better input (like Button A).

4. The Gallery Walk & Reflection (10 Minutes)

The Gallery Walk: Teams demonstrate their code "in action" (strapped to a wrist or held in a hand).
Closing Discussion: "What was harder: building with cardboard (Lesson 1) or building with code? Why is **Troubleshooting** so important when you can't physically 'tape' a fix onto your code?"

🛠️ Materials Needed

Hardware: 1 micro:bit per pair (with battery pack).
Software: Laptop/Tablet with access to the MakeCode Editor.
The "Designer Notebook": To track **Test Results** and **Actions Required** (as seen in the student notes).

💡 Connection to Computer Science

Wrap up the lesson by explaining:

"Design Thinking is **non-linear**. Today, when your code didn't work and you went back to change your 'Idea,' you were practicing the same cycle professional software engineers use. You moved from **Testing** back to **Ideating** instantly!"

Gr 4 CS Chapter 1 Scratch Lesson Plan

Big Idea: Computer Science uses software like Scratch to create interactive solutions that respond to a user's feelings or needs.

🎯 Learning Objectives

Students will apply the **Design Thinking Process** to software development.
Students will understand how to use **Events** and **Looks** blocks to communicate an idea.
Students will practice **Troubleshooting** by gathering user feedback on their digital prototype.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Mind-Reading Machine (10 Minutes)

The Demo: Open a blank Scratch project. Ask: "Can a cat on a screen tell if you are happy or sad?"
The Concept: Explain that computers only know what we tell them. To make a program "empathize," we have to code it to listen to the user.
The Connection: Today, we are moving from physical prototypes (Desk Buddy) and hardware (micro:bit) to **Software Prototyping**.

2. The Mission: The "Mood Booster" App (10 Minutes)

The Problem: Sometimes students feel stressed or bored during a long school day and need a 30-second "brain break."
The Goal: Create a Scratch program where a sprite asks the user how they feel and then provides a "solution" (a joke, a dance, or a calm color change).
The Constraint: Since this is a **low-fidelity digital prototype**, students should focus on the *interaction* (Back-and-forth) rather than complex game physics.

3. Hands-On: Scratch Prototyping (30 Minutes)

Students work in pairs following the non-linear steps from their notes:

Understand & Define (5 mins): Partner A interviews Partner B: "What makes you laugh when you're bored? What helps you calm down when you're frustrated?"
Ideate & Plan (5 mins): Sketch a quick "Storyboard" in the **Designer Notebook** showing what the Sprite will say and what the user will click.
Create (20 mins): * Use the Sensing block: ask [How are you feeling?] and wait.
Use Control blocks: if <answer = bored> then...
Use Looks or Sound blocks to provide the "booster."
Analyze & Test: If the code doesn't work as planned, students must **Troubleshoot** by checking their logic (e.g., "Did I spell the answer correctly in the code?").

4. The Gallery Walk & Reflection (10 Minutes)

The Gallery Walk: Students leave their projects running. The class rotates, trying out different "Mood Boosters."

Closing Discussion: "When your user typed an answer you didn't expect, what part of the **Design Thinking Process** did you have to go back to?" (Usually *Planning* or *Troubleshooting*).

🛠️ Materials Needed

Hardware: Laptops or Tablets.
Software: Scratch (Online or Desktop version).
The "Designer Notebook": To draw the "Storyboard" (a digital version of a sketch).

💡 Connection to Computer Science

Wrap up the lesson by explaining:

"In your notes, it says Design Thinking is **non-linear**. Today, you might have started coding, realized your user wanted something else, and went back to **Empathizing**. That's exactly how professional app developers at Google or YouTube work—they 'Test' and 'Iterate' until the user is happy!"

Gr 4 CS Chapter 1 Unplugged Lesson Plan

Big Idea: Computer Science isn't just about code; it's about designing solutions for people.

🎯 Learning Objectives

Students will identify that every man-made object was once an idea.

Students will practice the first step of Design Thinking: **Empathy** (understanding a user's needs).

Students will create a **low-fidelity prototype** using craft materials.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The "Invisible" Designer (10 Minutes)

The Game: Ask students to pick one object they have with them (a shoe, a pencil case, a water bottle).

The Question: Ask, "If you could change one thing about this to make it better for a five-year-old, what would it be?".

The Reveal: Explain that everything around them—including the apps on a tablet—started with someone asking that exact question. **Design Thinking** is the "superpower" we use to turn those ideas into reality.

2. The Mission: The "Desk Buddy" Challenge (10 Minutes)

The Problem: Many students lose their pencils, their tablets tip over, or they have nowhere to put their "fidgets" while they work.

The Goal: Design a **"Desk Buddy"**—a physical tool that solves a specific problem for a classmate.

The Twist: They aren't designing for themselves. They must **interview** their partner first to gain an empathetic understanding of their needs.

3. Hands-On: Rapid Prototyping (30 Minutes)

Break the students into pairs. They follow these lightning-fast steps:

Interview (5 mins): Partner A asks Partner B, "What is the most annoying thing about your desk area?".

Sketch (5 mins): Partner A draws a quick "brainstorm" or **mind map** of a tool to fix that problem.

Build (20 mins): Using a "Maker Bin," students build a 3D model of their invention.

Note: Remind them it doesn't have to be pretty; the purpose is to **fail fast** and show the idea.

4. The Gallery Walk & Reflection (10 Minutes)

The Gallery Walk: Students place their "Desk Buddy" on their desks while the class walks around to see the different solutions.

Closing Discussion: "How did knowing what your partner needed change what you built?".

🛠️ Materials Needed

The "Maker Bin": Cardboard, index cards, masking tape, pipe cleaners, rubber bands, and scissors.

The "Designer Notebook": A simple piece of paper folded in half for sketching.

💡 Connection to Computer Science

Wrap up the lesson by explaining: "In our next few lessons, we are going to use this same **non-linear process** to design things on the computer. Before a programmer writes a single line of code for a game or an app, they have to do exactly what you did today: understand the user and build a prototype!".

Gr 4 CS Chapter 2 micro:bit Lesson Plan

Big Idea: Using hardware feedback and sensors to create an **accessible** device that doesn't rely on sight.

🎯 Learning Objectives

Students will apply the **Feedback Loop** to a hardware project.
Students will distinguish between **Visual Feedback** and **Physical/Auditory Feedback**.
Students will troubleshoot a micro:bit program to make it more **inclusive** for users with visual impairments.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Silent Alarm (10 Minutes)

The Scenario: Imagine you have a micro:bit alarm that shows a "SMILE" on the screen when it's time to wake up.
The Problem: What if the user is blind or has their eyes closed?
The Question: How can the micro:bit give "Feedback" that doesn't use the screen? (Buttons, sound, or shaking/tilting).
The Lesson: In CS, we must design for all senses, not just sight.

2. The Mission: The "Pocket Compass" (10 Minutes)

The Goal: Design a device that tells a user if they are tilted too far to the left or right without them looking at the LEDs.
The Constraint: You must use the **Accelerometer** (tilt sensor) and provide feedback using **Sound** or **Icons** (but pretend the screen is invisible).
The Focus: You will gather **Feedback** from a partner to see if your "No-Look" signals actually make sense.

3. Hands-On: Hardware Iteration (30 Minutes)

Students work in pairs to prototype and refine:

Design (10 mins): Code a simple tilt alarm.
Example: If tilted left -> play melody; If tilted right -> play different melody.
Test & Get Feedback (10 mins): Partner A closes their eyes. Partner B tilts the micro:bit.
Feedback Question: "Can you tell which way it is tilting just by the sound? Is the sound too annoying or too quiet?"
Revise (10 mins): Use the feedback to change the pitch, volume, or add a "Shake" vibration (if using V2) to make the feedback clearer.

4. The Gallery Walk & Reflection (10 Minutes)

The Gallery Walk: One student from each team wears a blindfold while the other guides them using only their micro:bit's feedback signals.

Closing Discussion: "Which was more effective: sound feedback or vibration? Why is it important to **Test Again** after you make a change?"

🛠️ Materials Needed

Hardware: 1 micro:bit per pair (V2 is best for sound/vibration, V1 requires headphones/alligator clips).
Software: MakeCode Editor.
The "Designer Notebook": To track the **Feedback Loop** stages (Design -> Test -> Revise).

💡 Connection to Computer Science

Explain to the class: "When you added a sound to your tilt sensor, you were improving the **Accessibility** of your code. Professional engineers do this so that people can use technology while driving (audio GPS) or if they have a disability. Feedback makes tech usable for *everyone*."

Gr 4 CS Chapter 2 Scratch Lesson Plan

Big Idea: Using the **Feedback Loop** to ensure a digital story or game is easy for anyone to navigate (Usability).

🎯 Learning Objectives

Students will identify **Usability** issues in a Scratch project (e.g., buttons too small, instructions unclear).
Students will use **Conditional Logic** (if-then) to respond to user errors.
Students will demonstrate how **Iteration** leads to a more professional final product.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The "Broken" Game (10 Minutes)

The Demo: Show a Scratch project where the "Start" button is tiny, hidden in a corner, and the instructions disappear in 1 second.
The Discussion: "Is this a good design? Why not?"
The Term: Introduce **Usability**. If a user can't figure out how to play your game, it doesn't matter how cool the code is!

2. The Mission: The "Tutorial" Challenge (10 Minutes)

The Problem: Many users get frustrated with new software because they don't know what to click.
The Goal: Create a 3-scene Scratch "Interactive Tour" that uses **Symbols** and **Text** to guide the user.
The Constraint: Your project must be "fail-proof." If a user clicks the wrong thing, the sprite should give **Feedback** to help them find the right path.

3. Hands-On: The Usability Test (30 Minutes)

Create (15 mins): Students code a simple "Click the Sprite" game.
Accessibility Tip: Use a sound AND a costume change when the sprite is clicked.
Get Feedback (5 mins): A "Guest User" from another team tries the game.
The Observer: The creator *cannot talk*. They must just watch where the user struggles.
Revise (10 mins): Based on where the user hesitated, students make the buttons bigger, the text clearer, or add helpful hints.

4. The Gallery Walk & Reflection (10 Minutes)

The Gallery Walk: Students play each other's "improved" versions.
Closing Discussion: "What was the most common feedback you got? Did anyone have to go back to the **Planning** stage to fix a major confusing part?"

🛠️ Materials Needed

Hardware: Laptops/Tablets.
Software: Scratch.
The "Designer Notebook": Use the "Action Required" table from the student notes to record bugs and

fixes.

💡 Connection to Computer Science

Wrap up the unit: "You've used the **Design Thinking Process** with cardboard, hardware, and software. You've learned that **Feedback** isn't a critique of you—it's a tool to make your work better. Whether you are building a desk or a website, always keep the **User** at the center of your loop!"

Gr 4 CS Chapter 2 Unplugged Lesson Plan

Big Idea: Feedback is the "secret sauce" that helps designers see what they missed, especially when it comes to **accessibility**.

🎯 Learning Objectives

Students will define **feedback** as information about how well a design is working.

Students will identify the difference between **usability** (ease of use) and **accessibility** (can everyone use it?).

Students will practice the **Feedback Loop** by revising a design based on a user's specific needs.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The "Mystery Box" (10 Minutes)

The Activity: Place a common object (like a stapler or a specific toy) inside a cardboard box with a small hole.
The Challenge: Have one student try to use the object without looking, while the rest of the class watches.
The Reveal: Ask the student, "What was hard about that?" and ask the class, "What could we change to make it easier for someone who can't see?"
The Lesson: This introduces **accessibility**—designing so people with different abilities can use a product.

2. The Mission: The "Universal Snack Grabber" (10 Minutes)

The Problem: A vending machine company wants a new "grabber" tool, but they forgot to test it with people who have different needs.

The Goal: Design a tool that can pick up a small ball or block.
The Twist: Your design must work for a "User" who can only use one hand or a "User" who is wearing a blindfold.

3. Hands-On: The Feedback Loop (30 Minutes)

Students work in pairs through the **Feedback Loop**:

Phase A: Design & Prototype (10 mins): Build a first version of the grabber using "Maker Bin" materials.

Phase B: Test & Get Feedback (10 mins): One partner acts as the "Accessibility Tester" (e.g., they close their eyes or tuck one arm behind their back) and tries the tool.

Example Feedback: "The handle is too slippery," or "I can't feel where the claw is."

Phase C: Revise (10 mins): Students use that feedback to change and improve their design (e.g., adding a textured grip or a bell that rings when the claw closes).

4. Reflection: Accessibility vs. Usability (10 Minutes)

The Discussion: "Was your tool easy to use at first (**usability**)? What did you change to make it so *everyone* could use it (**accessibility**)?"

Closing Thought: Feedback isn't about being "wrong"; it's about making your design more inclusive for everyone.

🛠️ Materials Needed

The "Maker Bin": Cardboard tubes, rubber bands, clothespins, string, and tape.
Testing Props: Blindfolds (or clean sleep masks) and scarves (to tie one arm back).
The "Designer Notebook": To record "What I heard" and "What I changed".

💡 Connection to Computer Science

Explain to the class: "Software designers use this same loop! If an app only uses red and green buttons, someone who is colorblind might struggle. Developers listen to that **feedback** and add symbols or text so the app is **accessible** to everyone."

Gr 4 CS Chapter 3 micro:bit Lesson Plan

Big Idea: Digital artifacts like **programs** are made of smaller artifacts called **algorithms**.

🎯 Learning Objectives

Students will create a digital **program** artifact on the micro:bit.
Students will use an **experiment** (a type of artifact) to test their code.
Students will see how a digital artifact can control a physical interaction.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Digital Handshake (10 Minutes)

The Challenge: How can we send a secret message across the room without talking or moving?
The Solution: We can create a digital artifact—a communication program.

2. The Mission: The "Morse Code" Messenger (10 Minutes)

The Goal: Design a program where pressing Button A sends a short flash (dot) and Button B sends a long flash (dash).
The Artifacts: Students will create two things:
The **Program** (the code).
A **Code Key** (a physical artifact/cheat sheet) so their partner knows what the flashes mean.

3. Hands-On: Coding the Artifact (30 Minutes)

Step 1: Ideate (5 mins): Decide which symbols represent which letters (e.g., Dot-Dot = "Hello").
Step 2: Create (15 mins): Use MakeCode to create the flashes.
Example Code: On Button A click -> Show Icon (Small Square) -> Pause 100ms -> Clear Screen.
Step 3: The Experiment (10 mins): Put one student at the back of the room and one at the front. Try to send a "Secret Artifact" (the message) using the micro:bit.

4. Reflection: Invisible Artifacts (10 Minutes)

Discussion: "You can't touch the code inside the micro:bit. Does that still make it an artifact?"
Closing: Refer to the notes—**Programs** and **Algorithms** are conceptual and digital artifacts that are just as real as a chair or a Lego brick!

Gr 4 CS Chapter 3 Scratch Lesson Plan

Big Idea: Software is a collection of artifacts (images, sounds, and code) working together.

🎯 Learning Objectives

Students will identify different artifacts within a Scratch project (Sprites, Backdrops, Scripts).
Students will conduct an **experiment** to see how users interact with their artifact.
Students will understand that a "Finished Program" is the final artifact of the design process.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: What's in a Game? (10 Minutes)

The Activity: Show a popular game (like Minecraft or Mario).
The Question: "How many different things had to be 'made' for this game to exist?"
The List: Music (Sound Artifact), Characters (Object Artifacts), Rules (Algorithm Artifacts).

2. The Mission: The "Interactive Exhibit" (10 Minutes)

The Goal: Create a Scratch project that acts as a "Digital Museum." When a user clicks on an item, the item must explain what kind of "Artifact" it is.
The Content: Students will create three sprites: a **Physical Object** (like a car), a **Blueprint** (like a house plan), and a **Code Block**.

3. Hands-On: Building the Museum (30 Minutes)

Design (10 mins): Students draw or choose sprites that represent different types of artifacts from their Chapter 3 notes.
Code (15 mins): Use the When this sprite clicked block.
Example: The "Car" sprite says, "I am a physical object artifact!"
Example: The "Script" sprite says, "I am an algorithm artifact!"
Test/Experiment (5 mins): Have a classmate click through the museum. Does the "Interactive Exhibit" (the program artifact) clearly teach the user?

4. Reflection: The Design Journey (10 Minutes)

Discussion: "Look at how far you've come. You started with a sketch (Prototype), wrote instructions (Algorithm), and finished with a working game (Program)."
Closing: "Every time you create in this class, you are an **Artifact Architect**!"

🛠️ Materials Needed for these Lessons:

Lesson 7: Plain paper, markers, and "Blueprint" worksheets.
Lesson 8: Micro:bits and "Secret Code Key" cards.
Lesson 9: Laptops/Tablets with Scratch.

Gr 4 CS Chapter 3 Unplugged Lesson Plan

Big Idea: An **artifact** is anything created through a design process, ranging from physical objects to step-by-step instructions.

🎯 Learning Objectives

Students will define an **artifact** and distinguish between physical, digital, and conceptual types.
Students will understand that **algorithms** are "recipe" artifacts used to solve problems.
Students will create a physical artifact based on a conceptual artifact (a plan).

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Time Capsule (10 Minutes)

The Activity: Show a "Mystery Bag" containing a spoon, a printed photo, and a handwritten recipe.
The Question: "If an archaeologist found these 1,000 years from now, what would they call them?"
The Reveal: Introduce the term **Artifact**. Explain that in Computer Science, artifacts aren't just old pots; they are the things *we* create, like code, apps, and plans.

2. The Mission: The "Paper Plane Factory" (10 Minutes)

The Concept: Explore the three types of artifacts mentioned in the notes: **Objects**, **Prototypes**, and **Algorithms**.
The Goal: One student acts as the "Architect" (creates the plan/algorithm) and the other acts as the "Builder" (creates the physical object).

3. Hands-On: From Plan to Product (30 Minutes)

Phase 1: The Conceptual Artifact (10 mins): Partners design a "Stunt Plane." They cannot touch the paper yet! They must write a 5-step **Algorithm** (set of instructions) and draw a **Blueprint** (Plan).
Phase 2: The Physical Artifact (10 mins): Partners swap instructions with another team. They must follow the instructions *exactly* to build the plane.
Phase 3: The Experiment (10 mins): Teams test the planes. If the plane doesn't fly, was the "Object" broken, or was the "Algorithm" (the instructions) poorly designed?

4. Reflection: The Creator's Mindset (10 Minutes)

Discussion: "Which artifact was more important: the paper plane you can touch, or the instructions that told you how to make it?"
Closing: "In CS, we spend a lot of time designing the *instructions* (algorithms) so the *result* (the program) works perfectly."

Gr 4 CS Chapter 4 micro:bit Lesson Plan

Big Idea: Engineers use technology like sensors and AI to monitor and solve complex real-world problems.

🎯 Learning Objectives

Students will model a solution for a complex problem (traffic safety) using a micro:bit.
Students will understand how **data collection** helps in the **Testing** phase.
Students will iterate on their code based on "real-world" changing conditions.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Self-Driving Dilemma (10 Minutes)

The Scenario: Based on the Chapter 4 notes, talk about self-driving cars.
The Question: "How does a car 'see' a person in the rain?"
The Reveal: It uses sensors and algorithms. Today, you are the engineer designing a "Smart Crossing" sensor for a busy city.

2. The Mission: The "Safety First" Sensor (10 Minutes)

The Goal: Program the micro:bit to be a "Pedestrian Detector."
The Requirement: If a person is nearby (Button A), the car must slow down. If the road is clear, the car can go.
The Twist: You must add a "Night Mode" (Light Sensor) because the problem changes when it's dark!

3. Hands-On: Prototyping the AI (30 Minutes)

Coding (20 mins):
Basic: On Button A -> Show Icon (STOP).
Complex: Use the Light Level sensor. If Light < 50, then Show Icon (CAUTION) because it's harder for cars to see at night.
Testing & Data (10 mins): Move the micro:bits under a desk (to simulate night) or near a window (day).
Troubleshooting: Did your sensor react fast enough? Do you need to go back to **Ideate** to make the warning icon brighter?

4. Reflection: Scaling Up (10 Minutes)

Discussion: "We used one micro:bit. In a real city, how many sensors would you need? How would they talk to each other?"
Closing: This is how we solve the complex problem of transportation safety—one piece of code at a time.

Gr 4 CS Chapter 4 Scratch Lesson Plan

Big Idea: Designers use **Simulations** (like the Virtual Reality mentioned in the notes) to test expensive or dangerous ideas safely.

🎯 Learning Objectives

Students will create a **Digital Simulation** of a complex problem.
Students will use **Variables** to track data (e.g., how many people are helped).
Students will understand that the **Prototype** phase in Scratch allows for "unlimited" testing without wasting real-world resources.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: Why Simulate? (10 Minutes)

The Question: "If you want to test a new bridge, do you build the real bridge and drive a heavy truck over it immediately?"
The Answer: No! You use a computer simulation.
The Connection: Scientists use simulations to solve complex problems like climate change or medicine delivery.

2. The Mission: The "Mobile Clinic" Route (10 Minutes)

The Problem: Based on Case Study 1 in your notes, a Mobile Medical Clinic needs to reach 3 different towns.
The Goal: Create a Scratch simulation where a "Clinic Van" has to navigate to different sprites (towns).
The Constraint: The van has a "Fuel" variable. If it runs out of fuel, the problem isn't solved!

3. Hands-On: Building the Simulation (30 Minutes)

Setup: Create a backdrop with three "Remote Towns."
Code the Logic:
Create a variable called Patients Helped.
When Clinic touches Town1 -> Change Patients Helped by 10.
Iterate: After testing, students might realize the van moves too slow. They go back to the **Ideate** stage: "How can we make the van faster or the route shorter?"
Testing: Have a partner try to reach all towns before the timer (another variable) runs out.

4. Final Unit Reflection: You Are a Designer (10 Minutes)

The Big Wrap-Up: Look back at all 12 lessons.
The Question: "Which was your favorite artifact? Which complex problem do you want to solve when you grow up?"
Closing: Design Thinking is a loop that never truly ends. You are now equipped with the tools to change the world!

🛠️ Materials Needed:

Lesson 10: Large paper for Stakeholder Maps, markers.
Lesson 11: Micro:bits, battery packs, and a dark area (like under desks).
Lesson 12: Laptops/Tablets with Scratch.

Gr 4 CS Chapter 4 Unplugged Lesson Plan

Big Idea: Design Thinking isn't just for small fixes; it's a superpower used to tackle **Complex Problems** that affect millions of people.

🎯 Learning Objectives

Students will distinguish between a **Simple Problem** and a **Complex Problem**.
Students will practice the **Empathize** and **Define** stages for a large-scale issue.
Students will understand that complex problems have many "connected parts" and no single easy answer.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: Simple vs. Complex (10 Minutes)

The Sorting Game: Tape a line down the middle of the classroom. On one side, put a sign that says "Simple" (one clear fix). On the other, "Complex" (many parts/people involved).
The Challenge: Read out scenarios and have students jump to the correct side:
- "Your shoelace is untied." (Simple)
- "There is too much plastic trash in the ocean." (Complex)
- "You are hungry for a snack." (Simple)
- "A town has no clean drinking water." (Complex)

The Lesson: Complex problems are big, but the Design Thinking steps stay the same!

2. The Mission: The "Eco-School" Challenge (10 Minutes)

The Problem: Our school produces a lot of waste (food, paper, plastic). It affects the janitors, the students, the budget, and the environment.
The Goal: Use the first two steps of Design Thinking to tackle this complex issue.
The Tool: A **Stakeholder Map** (identifying everyone affected).

3. Hands-On: Empathy Mapping (30 Minutes)

Step 1: Empathize (15 mins): Divide the class into groups. Each group "interviews" a different persona: The Janitor, The Principal, a Hungry Student, and a Squirrel in the playground.
Task: Write down what that person **Sees, Feels, and Does** regarding school waste.
Step 2: Define (15 mins): Create a "Big Picture" Problem Statement.
Example: "Our school needs a way to reduce lunchtime plastic because it makes extra work for janitors and hurts local wildlife."

4. Reflection: The Ripple Effect (10 Minutes)

Discussion: "Why couldn't we just say 'Ban all plastic' and be done with it?" (Because some students might need plastic straws for accessibility, or the cafeteria might not have a way to wash metal forks yet).
Closing: Complex problems require looking at *all* the parts before jumping to an idea.

Gr 4 CS Chapter 5 micro:bit Lesson Plan

Big Idea: We use code to save money and materials. Smart designs use sensors to reduce the **Total Cost** of running a system.

🎯 Learning Objectives

Students will understand how **Processing Costs** (like electricity) affect a design.
Students will program a micro:bit to be a "Smart Light" that saves energy.
Students will explain how "Automation" makes a design more **Affordable** over time.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Forgotten Light (10 Minutes)

The Question: "Who has ever left a light on in an empty room?"
The Cost: Explain that the "Material" here is electricity. If the light stays on all day, the **Total Cost** goes up.
The Solution: How can we use a micro:bit sensor to keep the cost down?

2. The Mission: The "Auto-Off" Desk Lamp (10 Minutes)

The Goal: Program the micro:bit to act as a lamp.
The Logic: The LEDs should only turn on if the room is dark (Light Sensor) AND if someone is there (Button A).
The Reasoning: This saves "Labor" (you don't have to remember to flip a switch) and "Processing Cost" (battery power).

3. Hands-On: Efficient Coding (30 Minutes)

Code (20 mins): * If Light Level < 50 AND Button A is pressed -> Show Icon (Lightbulb).
Else -> Clear Screen.
Testing the Efficiency (10 mins): Compare two micro:bits. One stays on forever; one uses the "Smart Code." Which battery will last longer?
Connection: This is why smart homes are "Suitable"—they do the job well while staying "Affordable."

4. Reflection: Software Costs (10 Minutes)

Discussion: "Does code cost money?"
Closing: Yes! The **Labor Cost** of the programmer. But a well-designed program saves thousands of dollars later by being efficient.

Gr 4 CS Chapter 5 Scratch Lesson Plan

Big Idea: In the digital world, "Materials" are things like storage space and art. "Cost" is determined by how much people will pay for your solution.

🎯 Learning Objectives

Students will simulate a "Design Budget" within a Scratch project.
Students will evaluate the **Suitability** of different digital materials (Sprites/Sounds).
Students will understand that **Shipping Costs** in tech often mean the time it takes to download an app.

1. The Hook: The 100GB Game (10 Minutes)

The Scenario: You want to download a new game, but it's so big it takes 3 days to finish.
The Problem: The designer used too many "Heavy" materials (huge 4K videos).
The Lesson: Even in Scratch, we have to choose materials that are **Available** (load quickly) and **Suitable** for our users.

2. The Mission: The "Budget Pet" Game (10 Minutes)

The Goal: Create a "Virtual Pet" game.
The Constraint: You have 100 "Coins" to build your game.
- Simple Sprite: 10 Coins
- Animated Sprite with 10 Costumes: 50 Coins
- Custom Recorded Sound: 30 Coins

3. Hands-On: Digital Trade-Offs (30 Minutes)

Planning (10 mins): Students pick their "Materials" from a list. They must balance high-quality animations with the total cost.
Creating (20 mins): Build the pet interaction. If they spent all their money on a fancy dragon but have no money left for "Food" code, is the design **Suitable**?
Iteration: Students "sell" their game idea to a partner. If the partner says it's too simple, the student must **Revise** their material choices.

4. Final Unit Wrap-Up (10 Minutes)

The Big Connection: "You've learned about Design Thinking, Feedback, Artifacts, Complex Problems, and now Cost/Materials. You are now officially Grade 4 Computer Scientists!"
Closing Activity: Students sign their "Designer Notebooks" as certified Junior Engineers.

Gr 4 CS Chapter 5 Unplugged Lesson Plan

Big Idea: Every design has a "hidden" side—the **cost** of the materials and how easy they are to find (**availability**).

🎯 Learning Objectives

Students will define **Availability** and **Cost** in the context of design.
Students will calculate the **Total Cost** of a project (Materials + Processing).
Students will make trade-offs between expensive, high-quality materials and affordable, available ones.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Gold-Plated Pencil (10 Minutes)

The Scenario: Hold up a regular wooden pencil. Ask: "If I designed a pencil made of solid gold that never broke, would it be a good design for our classroom?"
The Discussion: Students will likely say it's too expensive.
The Lesson: A design can be "perfect" on paper, but if it's not **affordable** or the materials aren't **available**, it can't be built. Today, you are the Project Managers!

2. The Mission: The "Emergency Shelter" (10 Minutes)

The Problem: A group of hikers is stuck in the woods. You need to design a small model shelter for them.
The Goal: Build the strongest shelter possible while staying under a **$50 Budget**.
The Constraint: Different materials have different "Price Tags" based on how rare they are.

3. Hands-On: The Material Shop (30 Minutes)

Step 1: The Catalog (5 mins): Hand out a "Price List":
- Index Cards (Available/Cheap): $2 each
- Straws (Available/Medium): $5 each
- Pipe Cleaners (Rare/Expensive): $15 each
- Tape (Processing Cost): $1 per inch

Step 2: Planning & Budgeting (10 mins): Before touching materials, students must write their "Recipe" and add up the **Total Cost**.
Step 3: Build & Revise (15 mins): Students "buy" their materials. If they go over budget, they must **Iterate**—can they swap an expensive pipe cleaner for two cheap index cards?

4. Reflection: The Sweet Spot (10 Minutes)

Discussion: "Did anyone find a material that was cheap but broke easily?" (Refer to the 'Thin Plastic' example in the notes).
Closing: A good designer finds the **"Sweet Spot"**: Available, Affordable, and Suitable!

Gr 4 CS Chapter 6 micro:bit Lesson Plan

Big Idea: A program is just a digital **sequence**. If the order is wrong, the logic fails.

🎯 Learning Objectives

Students will translate a real-world algorithm into a digital **sequence**.
Students will troubleshoot an algorithm where the steps are out of order.
Students will identify the **Input** (Button press) and **Output** (LED number).

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Mixed-Up Morning (10 Minutes)

The Scenario: You wake up, go to school, put on your shoes, then put on your socks.
The Question: "Does that work? Why not?"
The Connection: Computers follow a **Sequence**. Today, we are going to build a "Counting Algorithm" where the order is the most important part.

2. The Mission: The "Score Keeper" (10 Minutes)

The Goal: Create a micro:bit tool that counts up when you press A, but *only* if you follow a specific sequence.
The Algorithm: 1. Start at 0.
2. Add 1.
3. Show the number.

3. Hands-On: Coding the Sequence (30 Minutes)

Design (10 mins): Write the algorithm in the "Designer Notebook" first.
Code (15 mins): Build it in MakeCode.
The Bug Challenge: Intentionally put the Show Number block *before* the Change Variable by 1 block.
Test (5 mins): Press the button. Why is the number wrong? (Because the computer showed the old number before adding the new one!).
Fix: Move the blocks into the correct **Sequence**.

4. Reflection: Following the Flow (10 Minutes)

Discussion: "How did changing the order of the blocks change the result?"
Closing: Computers read code like a book—from top to bottom.

Gr 4 CS Chapter 6 Scratch Lesson Plan

Big Idea: Complex animations are just many small algorithms working together in a specific **Sequence**.

🎯 Learning Objectives

Students will create a visual algorithm in Scratch.
Students will use a **Process** (Move, Wait, Change Costume) to achieve a specific **Output**.
Students will visualize their algorithm using a **Storyboard**.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The Glitchy Dance (10 Minutes)

The Activity: The teacher tries to do a "Dance" but forgets the "Wait" step between moves. The result is a blurry, messy motion.
The Lesson: To make a sprite move smoothly, we need a precise algorithm with Wait blocks to give the user time to see each step.

2. The Mission: The "Sports Star" Animation (10 Minutes)

The Goal: Pick a sport (Soccer, Basketball, Dance). Create an algorithm that makes a Sprite perform a specific move (like kicking a ball).
The Steps: 1. Prepare (Input).
2. Move (Process).
3. Celebrate (Output).

3. Hands-On: Visualizing the Algorithm (30 Minutes)

Step 1: The Diagram (10 mins): Draw a 3-part diagram of the algorithm.
Box 1: Sprite starts at the ball.
Box 2: Sprite swings leg.
Box 3: Ball moves away.
Step 2: Coding the Sequence (20 mins): Translate the drawing into Scratch blocks.
Critical Thinking: If the ball moves *before* the foot touches it, is the algorithm good? (No, the sequence is wrong!).
Iteration: Adjust the "Wait" seconds to make the "Process" look realistic.

4. Final Reflection: The Language of Computers (10 Minutes)

Discussion: "Is an algorithm the same thing as code?"
The Answer: An algorithm is the *idea* (the plan); code is the *language* we use to tell the computer that plan.
Closing: You've now mastered the "Language of Computing." You are ready to build anything!

🛠️ Materials Needed:

Lesson 16: Paper, markers, and "sweatshirt" prop.
Lesson 17: Micro:bits and laptops.
Lesson 18: Laptops/Tablets with Scratch.

Would you like me to generate a "Recipe for a Task" worksheet for Lesson 16?

Gr 4 CS Chapter 6 Unplugged Lesson Plan

Big Idea: Computers are not "smart"—they are just extremely obedient. They need a perfect **sequence** of instructions to succeed.

🎯 Learning Objectives

Students will define an **algorithm** as a set of precise, step-by-step instructions.
Students will identify the three features of a good algorithm: **Input**, **Process**, and **Output**.
Students will understand why the **sequence** (order) of steps is critical.

⏱️ Lesson Outline (60 Minutes)

1. The Hook: The "literal" Robot (10 Minutes)

The Activity: The teacher acts as a "Robot" who wants to put on a sweatshirt.
The Challenge: Students give verbal instructions. If a student says "Put your arm in," the teacher puts their arm through the *neck* hole or the bottom.
The Lesson: "Wait a bit" or "Put it on" isn't precise enough for a computer. We need a better **Process**.

2. The Mission: The Algorithm Architect (10 Minutes)

The Concept: Review the "Toast" example from the notes.
The Goal: Write a "Recipe for a Task" that another student (the "Computer") can follow exactly without asking questions.
The Input: Identify the "ingredients" needed before starting.

3. Hands-On: The "Drawing Machine" (30 Minutes)

Step 1: The Input (5 mins): Partners choose their inputs (e.g., "1 piece of paper, 1 blue marker, 1 red marker").
Step 2: The Process (15 mins): Partner A (the Architect) writes an algorithm to draw a simple hidden picture (like a house or a smiley face).

> *Rule:* They cannot use the name of the object. They must use precise steps like: "Draw a 2-inch square in the center. Draw a triangle on top of the square."

Step 3: The Output (10 mins): Partner B (the Computer) follows the steps exactly.
The Reveal: Compare the intended drawing with the actual **Output**. Did the **Sequence** matter? What happens if you draw the roof *before* the house?

4. Reflection: Precision Matters (10 Minutes)

Discussion: "Where did the 'Computer' get confused? Was a step missing or out of order?"
Closing: In CS, if you miss one step, the computer crashes. Precision is the key!

Gr 4 CS Conclusion Lesson Plan (Final Project)

Big Idea: Students will use the full **Design Thinking Process** to create a functional **Artifact** (using Scratch, micro:bit, or both) that solves a **Complex Problem** in their school or community while considering **Materials** and **Algorithms**.

🎯 Learning Objectives

Students will navigate the **Non-Linear Design Process** from empathy to iteration.
Students will choose the most **Suitable** tool (Scratch or micro:bit) for their specific solution.
Students will present a final **Program Artifact** and a **Process Journal** (Project Documentation).

⏱️ Project Timeline (2–3 Class Periods)

1. Phase 1: Empathy & The "Complex" Definition (30 Minutes)

The Challenge: Identify a problem in the school (e.g., noisy hallways, forgotten water bottles, students feeling lonely at recess, or energy waste).
The Task: Teams must interview at least two "Stakeholders" (teachers, younger students, or office staff).
The Output: A finalized **Problem Statement** and a **Stakeholder Map**.

2. Phase 2: Ideating & Choosing Your "Materials" (30 Minutes)

The Decision: Teams must decide which "Material" is most **Available** and **Suitable** for their goal:
- **Micro:bit:** Best for physical problems (shaking, temperature, step counting, tilting).
- **Scratch:** Best for digital problems (visual stories, tutorials, surveys, or complex simulations).
- **Hybrid:** Using a micro:bit as a controller for a Scratch game!

The Budget Check: Students must list their "Costs"—how much time will the code take? Do they have the batteries or laptops needed?

3. Phase 3: The Algorithm & Build (60+ Minutes)

The Step-by-Step: Before coding, students must draw a **Flowchart** or write a 5-step **Algorithm** for their invention.
The Create Phase: Students build their prototype.
- *Example:* A "Recess Buddy" (Scratch) that suggests games to play.
- *Example:* A "Posture Protector" (micro:bit) that beeps if your backpack is tilted too much.

4. Phase 4: The Feedback Loop & Troubleshooting (30 Minutes)

The Testing Event: Teams swap projects. They use the "Test Result / Action Required" table from their notes.
The Iteration: Students must make at least **one change** based on user feedback (e.g., "The alarm was too quiet" or "The Scratch instructions were too fast").

5. The Grand Finale: The "Innovation Fair" (30 Minutes)

The Presentation: Students don't just show the toy; they explain the **Process**.

The Pitch: 1. "We **empathized** with [User]..."
2. "We built this **artifact** because..."
3. "Our **algorithm** works by..."
4. "One piece of **feedback** we used was..."

🛠️ Materials Needed

Hardware: Micro:bits, battery packs, laptops/tablets.
Crafts: (Optional) Cardboard to create a "housing" for the micro:bit.
Documentation: A "Final Project Portfolio" (can be a folder with all their sketches and notes from the unit).

💡 Final Connection to Computer Science

Wrap up the entire unit by telling the students:

"You started this journey thinking Computer Science was just typing on a screen. Today, you proved it is about **people**. You looked at a human problem, planned a logical solution, listened to feedback, and built something that didn't exist before. You aren't just students anymore—you are **Designers and Engineers**."

Would you like a "Final Project Rubric" to help grade their Empathy, Coding, and Iteration steps?