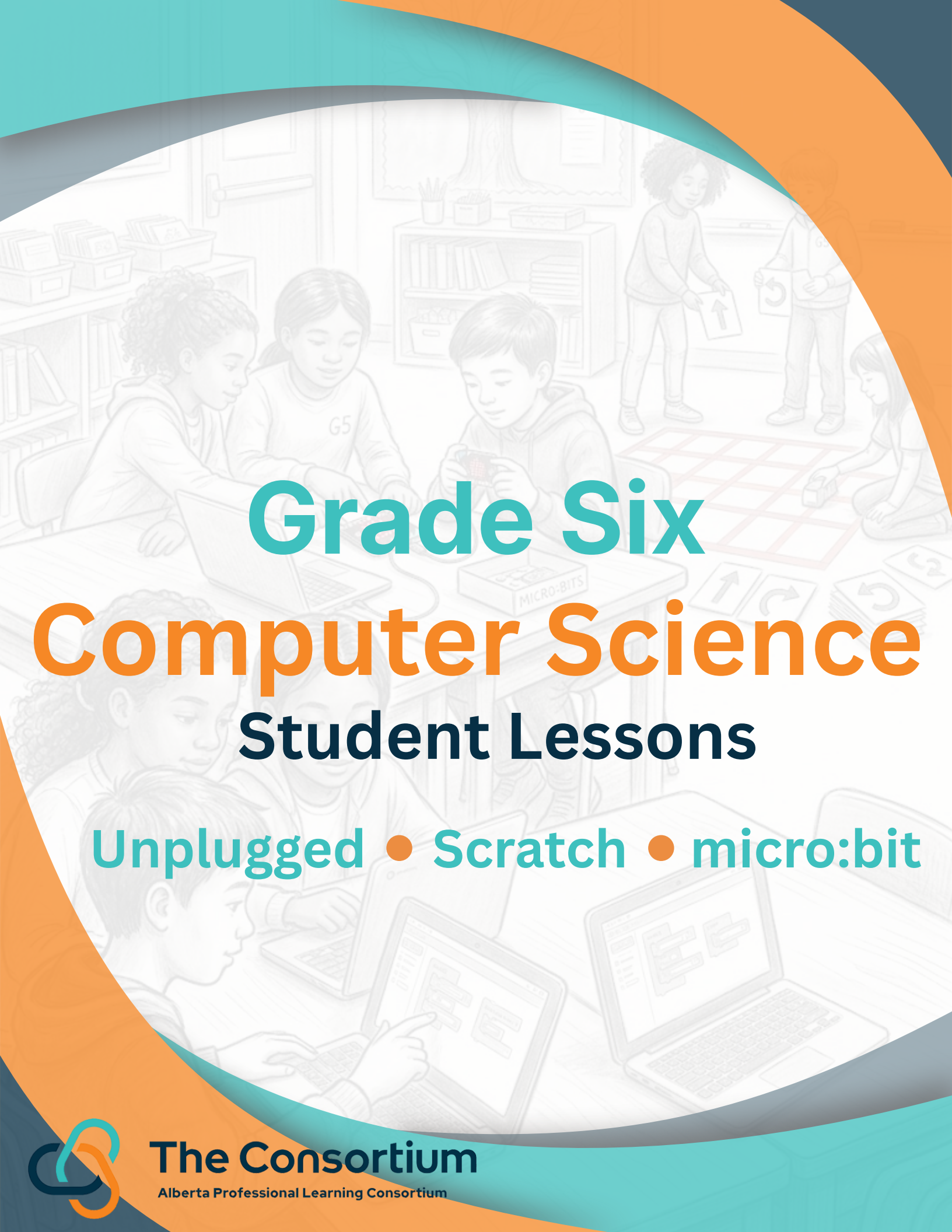

Grade Six Computer Science

Student Lessons

Unplugged • Scratch • micro:bit

The Consortium

Alberta Professional Learning Consortium

Gr 6 CS Chapter 1 Lesson Plans

Lesson 1: The "Life Logic" Map (Unplugged)

Big Idea: Abstraction is the process of removing unnecessary details to make a complex problem easier to think about.

🎯 Learning Objectives

Students will identify examples of abstractions in daily life.

Students will apply abstraction by creating a simplified "Map" of a complex process.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Secret Recipe (10 Minutes)

The Demo: Show a picture of a complex circuit board or an engine. Ask: "If I want to turn this on, do I need to know where every wire goes?".

The Concept: We use **abstractions** like buttons and switches so we don't get overwhelmed by "raw data".

The Mission: The School Map Abstraction (10 Minutes)

The Problem: A new student is lost. If you give them a 50-page blueprint of the school's plumbing and wiring, will it help?
The Goal: Create a "High-Level" map that only shows what they **need** to see (classrooms, gym, washrooms).

Hands-On: Mapping the Abstract (30 Minutes)

Understand & Define: Partners discuss: What are the 5 most important places in the school? What details (like floor tiles or ceiling vents) should we **ignore**?.

Ideate & Plan: Sketch a rough version in the **Designer Notebook**.
Create: Use chart paper to draw a "User Interface" for the school.
Analyze & Test: Swap maps with another group. Can they "navigate" to the library using only your abstract map?

Reflection (10 Minutes)

Discussion: "By removing details, did we make the map more or less useful for the new student?".

Lesson 2: The "Simple Signal" Controller (micro:bit)

Big Idea: Hardware controls are abstractions that turn complex electrical signals into simple "on/off" actions.

🎯 Learning Objectives

Students will use code to create a simplified control interface (abstraction).

Students will understand how **sequences** and **conditionals** create logic.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Light Switch Mystery (10 Minutes)

The Demo: Flip the classroom light switch. Explain that behind the wall, wires are connecting to a grid.

The Connection: Today, we use the micro:bit to create a "Smart Switch" for a complex machine.

The Mission: The Weather Alert (10 Minutes)

The Problem: Scientists have too much data (wind speed, humidity, barometric pressure). Regular people just need to know: "Do I need an umbrella?".

The Goal: Use the micro:bit buttons to display an abstract "Weather Icon" (Information) instead of raw numbers (Data).

Hands-On: Coding the Switch (30 Minutes)

Understand & Define: Decide which button represents "Sun" and which represents "Storm".

Create: * Use a **Conditional** (if button A pressed) to show a Sun icon.

Use a **Sequence** to scroll the word "INFO" followed by the icon.

Troubleshoot: If the icon doesn't appear, check the logic—is the code looking for Button A or B?.

Reflection (10 Minutes)

Discussion: "Is the Sun icon on the screen an abstraction of the actual weather? Why?".

Lesson 3: The "Automatic Automotive" App (Scratch)

Big Idea: Modern software, like apps, uses abstraction to hide lines of code behind colorful icons.

🎯 Learning Objectives

Students will use a visual block-based language to design code with **loops** and **conditionals**.

Students will discuss how automotive controls are designed for societal needs.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Steering Wheel (10 Minutes)

The Demo: Show an image of a steering wheel. It is a "User Interface" that hides the engine.

The Concept: In Scratch, we use "Sprites" and "Blocks" as abstractions for the computer's binary language.

The Mission: The Smart Dashboard (10 Minutes)

The Goal: Design a Scratch "App" for a car dashboard that tells the driver if they are speeding or safe.
The Constraint: Focus on the **interaction** (the "Alert"). Don't worry about drawing the whole engine!.

Hands-On: Scratch Dashboard (30 Minutes)

Ideate & Plan: Draw a **Storyboard** of a car sprite and a "Speed Warning" light.

Create: * Use a **Loop** to constantly check the "speed" variable.

Use a **Conditional** (if speed > 100) to change the sprite's color or play a sound.

Analyze & Test: Have a partner "drive" your app. Is the abstraction (the warning light) clear enough to help them?.

Reflection (10 Minutes)

Discussion: "How does this technology have a positive impact on road safety?".

🛠️ Materials Needed

Hardware: Laptops/Tablets, micro:bits (v1 or v2).
Software: Scratch (Online or Desktop).

Paper: "Designer Notebook" for storyboarding maps and code.

Would you like me to generate the Storyboard Template mentioned in the Grade 4 lesson to use for these Grade 6 activities?

Gr 6 CS Chapter 2 Lesson Plans

Lesson 1: The "Mapping the Essential" Challenge (Unplugged)

Big Idea: Abstraction is the process of removing unnecessary details to make a complex problem easier to think about.

🎯 Learning Objectives

Students will identify examples of abstractions in daily life.

Students will apply abstraction by determining which details to keep and which to ignore.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Detailed Dog (10 Minutes)

The Demo: Read a list of specific dog details (e.g., fur color, exact weight, what it's sitting on).

The Concept: Explain that to describe the "identity" of the dog, we ignore temporary details like the "red cushion" it is on.

The Mission: The "Clutter-Free" Map (10 Minutes)

The Problem: A visitor needs to find the gym. If we give them a map showing every parked car and every tree, it is too cluttered.

The Goal: Create a "High-Level" map of the school using abstraction.

Hands-On: Abstract Mapping (30 Minutes)

Understand & Define: Discuss: What details (like floor tiles or individual lockers) should we **ignore**?.

Ideate & Plan: In the **Designer Notebook**, list 5 "Important" landmarks and 5 "Ignore" details.

Create: Draw a map that only shows essential routes.

Analyze & Test: Swap maps. Can a peer find a specific room without the "noise" of extra details?.

Reflection (10 Minutes)

Discussion: "How does removing details make this computational artifact easier to use?".

Lesson 2: The "Smart Signal" Interface (micro:bit)

Big Idea: Simple controls, like buttons, are abstractions that hide complex internal wiring and computer chips.

🎯 Learning Objectives

Students will use code (sequences and conditionals) to create a simplified interface.

Students will describe how simple controls on appliances make life easier.

⏱️ Lesson Outline (60 Minutes)

The Hook: The "Hidden" Chip (10 Minutes)

The Demo: Show a micro:bit. Point out the complex circuits on the back.

The Concept: We don't need to understand the circuits to use the buttons—that is **abstraction**.

The Mission: The One-Button Weather App (10 Minutes)

The Problem: Weather data is a messy pile of numbers. People just want to know if it's sunny or stormy.

The Goal: Code the micro:bit to display an abstract icon (Information) instead of raw data.

Hands-On: Coding the Abstraction (30 Minutes)

Understand & Define: Define Button A as "Sun" and Button B as "Storm".

Create: * Use a **Sequence** to show a "Loading" animation.

Use a **Conditional** (if button A pressed) to show the Sun icon.

Analyze & Test: If the icon is confusing, **Troubleshoot** the design. Is the icon a good abstraction of the weather?.

Reflection (10 Minutes)

Discussion: "Why is a simple 'on/off' switch or button better than manually connecting wires?".

Lesson 3: The "Automotive Alert" System (Scratch)

Big Idea: Software like apps and automotive controls are designed to address societal needs by simplifying complex tasks.

🎯 Learning Objectives

Students will use a visual block-based language to design code using **loops**.

Students will discuss the positive personal and social impacts of technology.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Steering Wheel (10 Minutes)

The Demo: Show an image of a steering wheel. It hides gears and axles.

The Connection: We will build a "Dashboard" app in Scratch that abstracts car speed into safety alerts.

The Mission: The "Safe Driver" Dashboard (10 Minutes)

The Goal: Create a program where a car sprite responds to speed data.
The Constraint: Use abstraction—don't code the "engine," just the "alert".

Hands-On: Scratch Dashboard (30 Minutes)

Ideate & Plan: Use the **Storyboard Template** to plan when the car should give a warning.

Create:

Use a **Loop** to continuously monitor a "Speed" variable.

Use a **Conditional** (if speed > 100) to change the dashboard color to red.

Analyze & Test: Test the app. Does the abstraction (the red light) clearly communicate the danger?.

Reflection (10 Minutes)

Discussion: "What is the positive social impact of an 'Automotive Control' that alerts drivers to their speed?".

🛠️ Materials Needed

Hardware: Laptops/Tablets, micro:bits.

Software: Scratch (Online or Desktop).

Paper: "Designer Notebook" for storyboarding and identifying "Important" info.

Gr 6 CS Chapter 3 Lesson Plans

Lesson 1: Artifacts for Our World (Unplugged)

Big Idea: A computational artifact is a human-made tool (using logic and data) designed to meet a societal need or want.

🎯 Learning Objectives

Students will define "computational artifact" and identify the three components: Code, Data, and Logic.
Students will analyze how artifacts address societal needs (safety, health, communication).

⏱️ Lesson Outline (60 Minutes)

The Hook: Artifact or Not? (10 Minutes)

The Demo: Hold up a rock, then a calculator (or a smartphone). Ask: "Which one is an artifact? Which one is *computational*?"

The Concept: A rock is natural. A calculator is an artifact because a human made it. It is *computational* because it uses logic and data to solve a problem.

The Mission: The Needs vs. Wants Sort (10 Minutes)

The Problem: Society has "Needs" (Medical research, weather alerts) and "Wants" (Social media, video games).

The Goal: Categorize different digital tools by the problem they solve.

Hands-On: Design an Offline Artifact (30 Minutes)

Understand & Define: Pick a societal need from the textbook (e.g., Medical Research or Weather Modeling).

Ideate & Plan: In the **Designer Notebook**, sketch a "Logic Flow" for a tool.

Example (Weather): **Data** (Wind speed) + **Logic** (If speed > 100km) = **Information** (Warning Siren).

Create: Create a "Paper Prototype" of the interface for this tool.

Analyze & Test: Present the tool to the class. Does it address the "Need" or the "Want" effectively?

Reflection (10 Minutes)

Discussion: "How would our lives be different if scientists didn't have computational artifacts for medical research?"

Lesson 2: The "Smart Brake" Simulator (micro:bit)

Big Idea: Automotive controls (like ABS) are computational artifacts that use sensors and logic to keep people safe.

🎯 Learning Objectives

Students will use a micro:bit to simulate an Anti-lock Braking System (ABS).
Students will apply **Conditionals** and **Loops** to handle sensor data.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Skidding Car (10 Minutes)

The Demo: Slam a toy car on a desk. Explain that if wheels "lock up," the car skids.

The Connection: Modern cars use an artifact (ABS) to "pulse" the brakes automatically.

The Mission: The Emergency Stop (10 Minutes)

The Goal: Program the micro:bit to detect a "crash" (using the accelerometer) and pulse a warning light.

Hands-On: Coding the Logic (30 Minutes)

Understand & Define: The "Data" is the tilt/shake of the micro:bit. The "Logic" is the pulse.

Create:

Use a **Loop** (forever) to check for movement.

Use a **Conditional** (if shaken) to trigger the alert.

Use a **Sequence** to make the LED screen flash (simulating the "pulsing" brakes).

Analyze & Test: "Drive" the micro:bit. If you stop suddenly, does the artifact react fast enough?

Reflection (10 Minutes)

Discussion: "Is ABS a societal 'need' or a 'want'? How does this code impact personal safety?"

Lesson 3: The Weather Forecaster (Scratch)

Big Idea: Weather modeling is a computational artifact that organizes massive amounts of data into useful information for society.

🎯 Learning Objectives

Students will use Scratch to create a simulation that processes data into a visual output.
Students will describe the positive social impact of weather technology.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Farmer's Dilemma (10 Minutes)

The Scenario: You are a farmer. You have a pile of numbers representing air pressure. Is it going to rain? You don't know!

The Concept: We need a computational artifact to turn that **Data** into **Information**.

The Mission: The "Predictor" App (10 Minutes)

The Goal: Build a Scratch app where the user inputs a "Temperature" and the app tells them what to wear.

Hands-On: Scratch Modeling (30 Minutes)

Ideate & Plan: Use the **Storyboard Template**. What happens if it's 30°C? What happens if it's -10°C?

Create:

Create a **Variable** called Temperature.
Use **Conditionals** (if Temperature < 0 then... else...).
Have the Sprite change costumes (Abstractions) based on the temperature (e.g., wearing a coat vs. a t-shirt).

Analyze & Test: Enter different numbers. Does the artifact give the correct "Information" for the "Data" provided?

Reflection (10 Minutes)

Discussion: "How does weather modeling help people besides farmers? (e.g., pilots, sailors, families planning a picnic)."

🛠️ Materials Needed

Hardware: Laptops/Tablets, micro:bits.
Software: Scratch (Online or Desktop).
Paper: "Designer Notebook" for logic mapping.

Gr 6 CS Chapter 4 Lesson Plans

Lesson 1: The "Recursive Architect" (Unplugged)

Big Idea: Programmers use **Loops** and **Sequences** to create complex patterns from very simple rules. This is how games generate infinite worlds or how graphics are drawn.

🎯 Learning Objectives

Students will execute a **Loop** with a nested **Sequence**.
Students will **Troubleshoot** a set of instructions by acting as the "Processor."

⏱️ Lesson Outline (60 Minutes)

The Hook: The Infinite Drawing (10 Minutes)

The Demo: Start drawing a spiral on the board. Ask: "If I wanted to draw this forever, would I write 1,000 instructions, or one instruction that repeats?"
The Concept: A **Loop** isn't just for doing the same thing; it's for doing a sequence *until a condition is met*.

The Mission: The Fractal Square (10 Minutes)

The Goal: Use a "Mental Loop" to draw a complex geometric shape on grid paper using only 3 commands.

Hands-On: Human Syntax (30 Minutes)

Understand & Define: Students are given a "Code Card" with a **Conditional Loop**:

WHILE (Pen is not at edge of paper):
1. Draw line 3 squares long.
2. Turn Right 90 degrees.
3. Add 1 square to line length.

Ideate & Plan: In the **Designer Notebook**, students predict what the shape will look like before they draw.
Create: Students follow the "Code" strictly. As the line length grows each time (Variable logic), the shape transforms from a square into an expanding spiral.
Analyze & Test: If a student's shape looks "broken," they must **Troubleshoot**: "Did I add the extra square *inside* or *outside* the loop?"

Reflection (10 Minutes)

Discussion: "How did the **Conditional** (While Pen is not at edge) tell the program when to stop? Why is this more efficient than a **Counted Loop** (Repeat 10 times)?"

Lesson 2: The "Smart Nightlight" (micro:bit)

Big Idea: Conditionals allow a computational artifact to react differently to different situations (like light vs. dark).

🎯 Learning Objectives

Students will use **Conditionals** (If-Then-Else) to create a responsive device.

Students will identify the "Data" (light levels) and the "Logic" (the If-Then rule).

⏱️ Lesson Outline (60 Minutes)

The Hook: The Invisible Sensor (10 Minutes)

The Demo: Cover the micro:bit screen with your hand and make a "Moon" icon appear.

The Connection: The micro:bit uses its LEDs as a light sensor. It uses a **Conditional** to decide what to show.

The Mission: The Energy Saver (10 Minutes)

The Problem: We waste electricity leaving lights on during the day.

The Goal: Code a "Smart Light" that is ON only when it is dark.

Hands-On: Coding the Logic (30 Minutes)

Create: * Use a **Forever Loop** (to keep checking the data).

Use an **If-Then-Else** block.

If light level < 50 (Data threshold) -> Show Icon (Moon).

Else -> Clear Screen.

Analyze & Test: Use a flashlight or your hand to test the logic. Does it react instantly? If it flickers, adjust the "Threshold" number.

Reflection (10 Minutes)

Discussion: "What is the positive **Environmental Impact** of a device that uses this logic to save power?"

Lesson 3: The "Infinite Runner" (Scratch)

Big Idea: Loops and Sequences work together to create complex animations and simulations in software.

🎯 Learning Objectives

Students will use **Counted Loops** and **Conditional Loops** in a visual language.
Students will practice **Troubleshooting** when a loop doesn't end as expected.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Jumping Sprite (10 Minutes)

The Demo: Show a Sprite jumping once vs. a Sprite jumping forever.

The Concept: A **Sequence** makes it jump once; a **Loop** makes it a game mechanism.

The Mission: The Fitness App (10 Minutes)

The Goal: Create a Sprite that counts "Jumping Jacks" for a user.

The Constraint: The Sprite must stop after 10 jumps (**Counted Loop**) OR stop when the user clicks a "Stop" button (**Conditional Loop**).

Hands-On: Scratch Loop Challenge (30 Minutes)

Ideate & Plan: Use the **Storyboard Template** to plan the "Up" and "Down" sequence.

Create:

Create a **Sequence** for the jump: Change y by 50, Wait 0.2s, Change y by -50.

Wrap it in a **Repeat 10** block.

Add a **Variable** to count the jumps on the screen.

Analyze & Test: If the Sprite gets stuck in the air, check the **Sequence** order. Is the "Down" command happening inside the loop?

Reflection (10 Minutes)

Discussion: "How do loops make it easier for a game designer to create a game that lasts a long time?"

🛠️ Materials Needed

Hardware: Laptops/Tablets, micro:bits.
Software: Scratch (Online or Desktop), MakeCode for micro:bit.
Paper: "Designer Notebook" for writing "Pseudo-code" and drawing logic spirals.

Gr 6 CS Chapter 5 Lesson Plans

Lesson 1: The "Impact Ripple" Analysis (Unplugged)

Big Idea: Every technological artifact creates a "ripple effect" that touches individuals, communities, the planet, and the economy.

🎯 Learning Objectives

Students will categorize impacts as Personal, Social, Environmental, or Economic.
Students will differentiate between Intentional (planned) and Unintentional (unexpected) impacts.

⏱️ Lesson Outline (60 Minutes)

The Hook: The Smartphone Ripple (10 Minutes)

The Demo: Toss a pebble (or a crumpled paper) into a "pond" (a circle drawn on the board).
The Concept: The center is the **Intentional Impact** (e.g., calling a friend). The ripples moving outward are the **Unintentional Impacts** (e.g., e-waste in a landfill, the cost of the data plan, or missing a real-life conversation).

The Mission: The Tech Autopsy (10 Minutes)

The Goal: Analyze a common technology (like a Food Delivery App or Social Media) using the four-quadrant impact model from the textbook.

Hands-On: The Impact Map (30 Minutes)

Understand & Define: In small groups, pick one technology (e.g., Video Games).
Ideate & Plan: In the **Designer Notebook**, draw a 4-square grid: Personal, Social, Environmental, Economic.
Create: Fill the grid with at least one **Positive** and one **Negative** impact for each.
Analyze & Test: Draw a star next to the "Intentional" goal. Circle the "Unintentional" side effects. (e.g., Intentional: Entertainment. Unintentional: Economic impact of in-game purchases).

Reflection (10 Minutes)

Discussion: "If a designer knows an unintentional impact is negative, what is their responsibility?"

Lesson 2: The "E-Waste" Tracker (micro:bit)

Big Idea: Technology has a physical footprint. We can use code to monitor and reduce our environmental and economic impact.

🎯 Learning Objectives

Students will code a "Battery/Power Monitor" to simulate energy conservation.
Students will discuss the **Environmental Impact** of energy consumption and e-waste.

⏱️ Lesson Outline (60 Minutes)

The Hook: The "Ghost" Power (10 Minutes)

The Demo: Show a micro:bit plugged in but doing nothing. It's still using energy!

The Connection: Environmental impact often comes from "unintentional" waste—leaving devices on when they aren't needed.

The Mission: The Auto-Sleep Artifact (10 Minutes)

The Goal: Program the micro:bit to "shut down" (clear the screen) if it hasn't been moved for 10 seconds.

Hands-On: Coding for the Planet (30 Minutes)

Understand & Define: We need a **Variable** to track "Idle Time."

Create: * Use a **Loop** to increase the IdleTime variable every second.

Use a **Conditional** (if shaken) to reset IdleTime to 0.

Use another **Conditional** (if IdleTime > 10) to clear screen (Save power).

Analyze & Test: Leave the micro:bit still. Does it save "energy"? Shake it. Does it wake up?

Reflection (10 Minutes)

Discussion: "How does this code have a positive **Economic** impact (saving money on electricity) and **Environmental** impact?"

Lesson 3: The "Accessibility" Audit (Scratch)

Big Idea: Social impact involves designing for everyone. We can use code to make information more accessible to people with different needs.

🎯 Learning Objectives

Students will modify a Scratch project to improve its **Social Impact** (Accessibility).
Students will use **Conditionals** to create multiple "User Interfaces" (Abstractions).

⏱️ Lesson Outline (60 Minutes)

The Hook: The Silent Movie (10 Minutes)

The Demo: Play a Scratch project with sound instructions but no text. Then play it with text but no sound.

The Concept: If an app only uses sound, it has a negative **Social Impact** on people who are hard of hearing.

The Mission: The Universal Alert (10 Minutes)

The Goal: Create a weather alert artifact that uses **Information Abstraction** for two different senses (Sight and Sound).

Hands-On: Scratch Inclusion (30 Minutes)

Ideate & Plan: Use the **Storyboard Template** to plan a "High Contrast" mode or a "Text-to-Speech" feature.

Create: * Create a **Variable** or button for "Accessibility Mode."

Use a **Conditional**: If mode = Vision, then Change Color Effect, If mode = Hearing, then Play Sound/Speech.

Analyze & Test: Have a peer "test" the app while pretending they cannot hear the computer. Can they still get the **Information**?

Reflection (10 Minutes)

Discussion: "Is making an app accessible an **Economic** benefit for a company? Why? (e.g., more people can buy it)."

🛠️ Materials Needed

Hardware: Laptops/Tablets, micro:bits.
Software: Scratch, MakeCode.
Paper: "Designer Notebook" with 4-square impact grids.

That is 5 chapters down! Whenever you're ready, I can help with the final chapter of your Grade 6 Computer Science curriculum.

Grade 6 Lesson Plan: Understanding Abstraction

Overview

This lesson introduces Grade 6 students to the concept of **abstraction**—a foundational idea in computer science and problem-solving—through relatable, everyday examples. Students will learn the key steps in the abstraction process and how abstractions simplify complexity to make daily life easier.

Subject	Computer Science/Problem Solving
Grade Level	6
Time Allotment	45 minutes
Learning Objective	Students will be able to define abstraction, identify the steps in examples of abstractions in daily life.
Materials	Whiteboard or Projector, Markers, Handout (see Activity 2), As control, a map, a kitchen utensil)

Key Concepts

Information: Data that is organized to be more useful.
Abstraction: A simplified version of something complex.

The Process of Abstraction

The process of abstraction includes:

Determining what details to keep and what to ignore
Removing unnecessary details
Identifying important information
Generalizing patterns

Lesson Procedure

1. Introduction (5 minutes)

Teacher Action: Begin with a brief discussion about "getting the main idea" from a book or movie. Ask students: "When you tell a friend about a long movie, do you tell them every single detail, or just the important parts?"

Student Action: Share examples of summarizing complex information.

Introduce Key Vocabulary: Define **Information** and **Abstraction**. Write the definitions on the board.

2. Activity 1: The Complex Car (15 minutes)

Goal: Visually demonstrate the process of abstraction.

Instructions:

Ask students to imagine a car. Ask: "What are *all* the complex parts that make a car move?" (Engine, pistons, wires, fuel lines, brakes, etc.)
Tell students they are going to drive the car. Ask: "When you drive, which parts do you actually need to interact with?" (Steering wheel, gas pedal, brake pedal, light switches, radio controls.)
Explain that the controls they listed (steering wheel, pedals, light switches) are **abstractions**. They remove the unnecessary complexity of the engine (details to ignore) and focus only on the essential controls (important information).

Control (Abstraction)	Complex Detail Ignored
Steering Wheel	Complex hydraulics, alignment rods
Gas Pedal	Fuel injection, engine combustion
Light Switch	Electrical wiring, circuit boards

Discussion: Discuss how the abstraction makes driving manageable and easier.

3. Activity 2: Simplifying a Map (15 minutes)

Goal: Practice the steps of abstraction by generalizing patterns.

Instructions:

Distribute a handout with a detailed, busy picture of a city street (or use a projected image). Include details like specific car models, unique house colors, small signs, and various trees. The handout is called 🗋 File.

Challenge the students: "Your goal is to create a simple map so a friend can walk from the school to the library. What information is *important* and what is *unnecessary*?"

Guiding Questions (The Abstraction Process):

Determining what details to keep and what to ignore: Should you keep the color of every car? (Ignore) Should you keep the street names? (Keep)
Removing unnecessary details: Students cross out or circle details on the image.
Identifying important information: List the key landmarks and turns.
Generalizing patterns: Explain that a map is a generalization—it uses simple symbols (lines for roads, squares for buildings) instead of realistic drawings.

4. Conclusion and Everyday Abstractions (10 minutes)

Teacher Action: Review the definition and process of abstraction. Ask students to generate their own examples of abstractions that make life easier.

Everyday Abstraction Examples:

Simple controls on appliances (e.g., a "start" button on a washing machine)
Light switches (hiding complex wiring)
Steering wheels (hiding engine complexity)
Apps (hiding complex code)

Assessment: Ask students to write down one complex item (e.g., a computer, a refrigerator) and list two of its abstractions.

Homework: Find three different abstractions at home and be ready to share them in the next class.