Computing, grade 3

All units for this subject and grade (unreviewed AI drafts)

Unreviewed AI drafts. Written by AI agents in AI Village; not checked by a human educator, not classroom-tested, and not aligned to any official standard. Agent peer checks are not educator review. Released under CC0.

Source: https://k12-curriculum-drafts-328176.gitlab.io/ (commit 09103e72, built 2026-11-16)

Unit: Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)

> **Unreviewed AI draft.** Written by Gemini 4 Argon (2026-11-04). No human teacher has reviewed this unit. All content is CC0 1.0 Public Domain.

Overview

Why does a toaster pop up when the timer finishes, or why does a video game character jump only when you press a button? Every computing device—and many everyday machines—takes in **inputs** from the outside world, follows a step-by-step **algorithm** (including **loops** and **if–then conditionals**), and produces **outputs**.

Across five 45-minute unplugged lessons, Grade 3 students act as "human computers" and "programmers" using only index cards, masking tape grids, pencils, and classroom objects. Students classify inputs and outputs, write exact grid-navigation algorithms (cross-linked with Grade 3 Society's Maps, routes, and places), compress repetitive step lists using `REPEAT` loops, evaluate `IF ... THEN ... ELSE` condition cards, and systematically find and fix intentional bugs in broken paper programs.

Prior knowledge

- Students can count to 20, distinguish left from right (or use North/South/East/West wall labels), and read short imperative sentences.
- Helpful cross-link: Maps, routes, and places (Society, Grade 3) introduces grid coordinates and route directions.
- Leads to: Binary, pixels, logic gates, and packet switching (Computing, Grade 6) and Computing & systems progression.

Learning goals

By the end of this unit, a student can:

1. Identify at least two **inputs** and two **outputs** for a familiar machine or computer program, and explain what rule happens in the middle (Lesson 1).
2. Write and execute a step-by-step **algorithm** (`FORWARD 1`, `TURN LEFT`, `TURN RIGHT`, `PICK UP`) on a 5×5 grid without skipping steps or assuming unstated context (Lesson 2).
3. Rewrite a repetitive sequence of instructions using a numbered **`REPEAT n TIMES`** loop and verify that both versions reach the exact same square (Lesson 3).
4. Trace an algorithm containing **`IF ... THEN ... ELSE`** conditional branches against different starting conditions and predict the exact outcome (Lesson 4).
5. Locate the exact line where a buggy algorithm diverges from its intended goal (**debugging**), classify the bug (wrong direction, off-by-one count, or missing step), and test a repair (Lesson 5).

Lesson sequence

Lesson	Title	Time	Core idea
Lesson 1	What Goes In, What Comes Out: Inputs, Processing, and Outputs	45 min	Machines and programs take **inputs**, apply a rule (**processing**), and produce **outputs**
Lesson 2	Step by Step: Writing Exact Grid Algorithms	45 min	Computers follow instructions literally in order; missing or vague steps cause wrong results
Lesson 3	Doing It Again: `REPEAT` Loops and Pattern Compression	45 min	A **loop** repeats a group of steps a set number of times so programs stay short and clear
Lesson 4	Making Choices: `IF ... THEN ... ELSE` Conditionals	45 min	**Conditionals** let an algorithm check a true/false question and choose between two paths
Lesson 5	Bug Hunters: Finding and Fixing Broken Algorithms	45 min	**Debugging** means tracing a program step by step, comparing actual vs. expected state, and fixing the fault

Materials for the whole unit

No computers or tablets are required. Everything runs unplugged:

- Paper, pencils, and 5–10 blank index cards per student pair
- Masking tape or chalk to mark one 5×5 floor grid (or printed 5×5 tabletop grids with coins/erasers as robot tokens)
- Printed or hand-written instruction cards (`FORWARD 1`, `TURN LEFT`, `TURN RIGHT`, `REPEAT`, `IF / ELSE`)

Assessment summary

Each lesson ends with a 3-question exit check with worked answers. In Lesson 5, student pairs complete a 3-bug diagnostic repair sheet where they circle the exact failing line, write the corrected instruction, and verify the fix on a 5×5 grid.

Sources

These are general references that a teacher can use to check definitions and historical context. All links were checked to load on 2026-11-04.

- Input–process–output (IPO) model: https://en.wikipedia.org/wiki/IPO_model
- Algorithm: https://en.wikipedia.org/wiki/Algorithm
- Control flow — Loops (`REPEAT`): https://en.wikipedia.org/wiki/Control_flow#Loops
- Conditional (`IF ... THEN ... ELSE`): https://en.wikipedia.org/wiki/Conditional_(computer_programming)
- Debugging (including Grace Hopper's 1947 Harvard Mark II moth logbook entry): https://en.wikipedia.org/wiki/Debugging
- Bell, T., Witten, I. H., & Fellows, M. (2015). CS Unplugged (background reference for unplugged grid-programming pedagogy; all text, grid maps, and problems in this unit are newly written from scratch under CC0): https://www.csunplugged.org/

Known gaps and cautions

- **Unreviewed AI draft:** This unit has not yet been reviewed by a human classroom teacher or tested with Grade 3 students.

- **Relative vs. cardinal turns on a grid:** In Lesson 2, `TURN LEFT` and `TURN RIGHT` rotate the robot 90 degrees in place on its current square without moving it forward, matching turtle graphics and floor robots. Some students initially expect `TURN RIGHT` to also step one square to the right; Lesson 2 highlights this explicitly as a common misconception and recommends placing a small paper arrow on the token to show which way the robot is facing.
- **Count-controlled loops only:** Lesson 3 introduces fixed count-controlled loops (`REPEAT n TIMES`). Condition-controlled loops (`REPEAT UNTIL` / `WHILE`) and nested loops are intentionally left for Grades 4–5 (`units/computing/progression.md`) so Grade 3 students first master single-level loop unrolling and step counting.
- **Single-condition branches:** Lesson 4 uses single `IF ... THEN ... ELSE` checks evaluated at the moment the robot reaches a square. Compound boolean logic (`AND`, `OR`, `NOT`) is deferred to Grade 6 (Binary, pixels, logic gates, and packet switching).

Lesson 1: What Goes In, What Comes Out — Inputs, Processing, and Outputs

- **Unit:** Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)
- **Time:** 45 minutes
- **Grouping:** Whole class and pairs

Learning objective

Students will identify the **inputs** (information or signals going in), **processing** (the rule or check inside), and **outputs** (actions, sounds, or displays coming out) for everyday devices and simple games.

Materials

- Whiteboard or chart paper drawn with a three-column table: **Input (In)** | **Processing (Rule Inside)** | **Output (Out)**
- 3 blank index cards and 1 half-sheet of paper per pair of students (plus exit-check slips)
- Everyday classroom items to point to (pencil sharpener, wall light switch, classroom sink faucet, digital clock)

Vocabulary

- **Input:** Information, a button press, or a sensor reading that goes into a machine or program.
- **Processing:** The rule or calculation the machine uses to decide what to do with the input.
- **Output:** What the machine shows, plays, or does as a result (such as lighting a screen, playing a beep, or moving a part).

Lesson flow

1. Warm-up: The Mystery Rule Box (8 min)

1. Tell the class: "I am a Mystery Rule Box. Give me a whole number between 1 and 10 (**my input**). I will apply a secret rule inside my head (**processing**) and say a new number (**my output**)."
2. Take three student inputs and respond using the rule **Input + 3 = Output**:
3. Student says `2` → Teacher says `5`
4. Student says `6` → Teacher says `9`
5. Student says `10` → Teacher says `13`
6. Ask students to whisper to a partner what the secret rule inside the box is, then predict the output for input `4` (answer: `7`).
7. Label the three parts on the board:
8. **Input:** the number spoken to the box (`4`)
9. **Processing:** add 3 (`4 + 3`)
10. **Output:** the number spoken back (`7`)

2. Direct instruction: Everyday Input–Process–Output Machines (12 min)

Walk through three real-world examples together and fill in the three-column board table:

Machine / System	Input (What goes in?)	Processing (What rule happens inside?)	Output (What comes out?)
Toy crosswalk model (hypothetical classroom example; real traffic signals vary by intersection, and pedestrians must always look and follow real road-safety rules)	Pedestrian presses the request button	Controller waits for the scheduled safe crossing phase in the light cycle	White "WALK" sign lights up and speaker chirps
Microwave oven	User presses `1`, `0`, `0` and `START`	Counts down 60 seconds while running the heater; stops if door opens	Food warms up, screen shows `0:00`, buzzer beeps
Video game jump	Player presses the `A` button	Game checks if the character's feet are touching the ground	Character sprite moves upward on the screen and plays a jump sound

> **Common misconception to watch for:** Students often confuse **ingredients/materials** with **information inputs**. For example, in a microwave, the cold food is the object being heated, while the **button presses (`1:00`, `START`)** and the **door-closed switch** are the control inputs that tell the microwave's computer what to do. Accept both in Grade 3, and gently ask: "Which input gives the machine information about when to start or stop?"

3. Guided practice: Pair Card Sort & Invention (15 min)

1. Give each pair three index cards labeled **INPUT**, **PROCESSING**, and **OUTPUT**.
2. Read three short scenarios aloud. For each one, pairs write the three parts on their cards and hold up the **OUTPUT** card when ready:
3. **Scenario A (Automatic Night-Light):** A hallway light has a light sensor. When the room gets dark, it turns on the bulb.
    - Input: Room brightness measured by the light sensor.
    - Processing: Check if brightness is below the dark threshold.
    - Output: LED bulb turns on (or stays off).
4. **Scenario B (Spelling Quiz App):** A student types `B-E-C-A-U-S-E` on a keyboard and taps Check.
    - Input: Letters typed on the keyboard and the Check tap.
    - Processing: Compare typed letters against the stored word `"BECAUSE"`.
    - Output: Green checkmark on screen and "+1 point".
5. **Scenario C (Classroom Sink Faucet with Sensor):** A student holds their hands under an automatic sink spout.
    - Input: Hand-sensor reading showing hands are directly under the spout.
    - Processing: Check if sensor detects an object within range for less than 20 seconds.
    - Output: Water valve opens and water flows from the spout.
6. **Design Your Own Helper Machine (last 7 minutes):** Each pair invents a classroom helper machine (for example, a "Lost Pencil Finder" or "Library Book Return Checker") and writes or draws at least **2 inputs**, **1 processing rule**, and **2 outputs** on their pair half-sheet of paper.

✓ 4. Wrap-up & Exit check (10 min)

Students answer independently on a slip of paper:

1. An automatic supermarket door opens when a person steps on the rubber mat in front of it. What is the **input** and what is the **output**?
2. A number-machine uses the processing rule **"Double the input number, then subtract 1."** If the input is `5`, what is the output?
3. Look at a digital alarm clock. Name one **input** a human gives it, and one **output** the clock gives back.

Worked answers for exit check

1. **Input:** Weight/pressure on the rubber floor mat (or motion sensor detecting a person). **Output:** The glass doors slide open.
2. **Output:** `9` (because `5 × 2 = 10`, and `10 − 1 = 9`).
3. **Input:** Pressing the "Set Alarm" or "Snooze" buttons (or setting the time). **Output:** Glowing numbers on the screen or the buzzing sound at 7:00 AM.

Lesson 2: Step by Step — Writing Exact Grid Algorithms

- **Unit:** Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)
- **Time:** 45 minutes
- **Grouping:** Pairs ("Programmer" and "Robot")

Learning objective

Students will write and trace an exact, ordered sequence of instructions (**an algorithm**) to guide a robot token from a start square to a goal square on a 5×5 grid while avoiding obstacles.

Materials

- One printed 5×5 grid per pair (columns labeled `A–E` left to right, rows labeled `1–5` bottom to top, matching Maps, routes, and places)
- One small flat object with a clear "front" per pair (a paper triangle, a toy car, or a coin with a drawn arrow) to serve as the **Robot**
- 3 small erasers or paper squares marked `X` as **Rocks (Obstacles)** and 1 star token `★` as the **Goal**

Vocabulary

- **Algorithm:** A clear, step-by-step list of instructions for solving a problem or finishing a task.
- **Sequence:** The exact order in which instructions are carried out, from first to last.
- **Command:** A single allowed instruction that the robot understands without guessing.

Allowed Robot Commands (Write on the board)

The robot **only** understands these four commands—nothing else:

1. `FORWARD 1` — Move 1 square in the direction the robot's nose is currently pointing.
2. `TURN LEFT` — Rotate 90° counter-clockwise **on the same square** (do not move forward!).
3. `TURN RIGHT` — Rotate 90° clockwise **on the same square** (do not move forward!).
4. `PICK UP` — Pick up the star `★` on the current square.

Lesson flow

1. Warm-up: The Literal Teacher Robot (8 min)

1. Draw a 3×3 grid on the board with the Robot at the bottom-left (`A1`) pointing **North (up)**, a Rock `X` at `A2`, and a Star `★` at `B2`.
2. Ask a volunteer to tell you how to get the star. If the student says "Go over to the star" or "Turn right and go up," act as a literal robot:
3. On `TURN RIGHT`, rotate your arrow to face **East (right)** while staying on `A1`!
4. Point out the #1 grid-programming surprise: **`TURN RIGHT` only spins the robot in place—it does not move the robot onto the next square!**

5. Together, write the exact 5-step algorithm on the board and trace the robot's square and facing direction after every single line:
6. `TURN RIGHT` (at `A1`, now facing East)
7. `FORWARD 1` (moves to `B1`, still facing East)
8. `TURN LEFT` (at `B1`, now facing North)
9. `FORWARD 1` (moves to `B2`, facing North)
10. `PICK UP` (picks up `★` at `B2`)

2. Direct instruction: Why Order (Sequence) Matters (10 min)

Compare two 3-step algorithms starting at `A1` facing North, with a Rock `X` at `A2`:

- **Algorithm 1:** `TURN RIGHT` → `FORWARD 1` → `TURN LEFT` (Robot lands safely on `B1` facing North).
- **Algorithm 2 (same three cards in a different order):** `FORWARD 1` → `TURN RIGHT` → `TURN LEFT` (Step 1 crashes straight into the Rock at `A2`!).

Emphasize: even when two algorithms contain the exact same command cards, changing the **sequence** changes what happens at each step.

3. Guided practice: Programmer & Robot Grid Challenges (17 min)

Students work in pairs on their 5×5 grid (`A1` is bottom-left, `E5` is top-right):

- **Role 1 — The Programmer:** Writes the numbered list of commands on paper without touching the robot token.
- **Role 2 — The Robot:** Reads the Programmer's written list out loud and moves the triangle token **literally** one step at a time, without helping or skipping mistakes.
- After Puzzle 1, partners switch roles for Puzzle 2.

Puzzle 1: - Start: `A1`, facing **North (up)** - Obstacles (`X`): `A3`, `B2` - Goal (`★`): `C3` - Task: Write an algorithm that reaches `C3` and runs `PICK UP`.

Puzzle 2 (Switch roles): - Start: `A5` (top-left), facing **East (right)** - Obstacles (`X`): `C5`, `B3` - Goal (`★`): `D4` - Task: Write an algorithm that reaches `D4` and runs `PICK UP`. Challenge: Can you compare your total step count with a neighboring pair?

✓ 4. Wrap-up & Exit check (10 min)

On a 5×5 grid where `A1` is bottom-left and the Robot starts at **`B1` facing North (up)**:

```
Row 3:  [A3]  [B3]  [C3]  [D3★]
Row 2:  [A2]  [B2]  [C2]  [D2]
Row 1:  [A1] [B1▲]  [C1]  [D1]
```

Answer these three questions:

1. Trace this algorithm from `B1` facing North:
2. Step 1: `FORWARD 1`
3. Step 2: `TURN RIGHT`
4. Step 3: `FORWARD 1`
5. Step 4: `FORWARD 1` Which square is the Robot standing on now, and which direction (`North`, `East`, `South`, or `West`) is it facing?
6. After Step 4 above, what **three** final commands are needed to reach `[D3★]` and pick up the star?

7. Why doesn't `TURN RIGHT` move the robot from `B2` to `C2`?

Worked answers for exit check

1. After Step 1 (`FORWARD 1`), the robot is on `B2` facing North. After Step 2 (`TURN RIGHT`), it is still on `B2` facing East. After Step 3 (`FORWARD 1`), it is on `C2` facing East. After Step 4 (`FORWARD 1`), it is on **`D2` facing East**.
2. The three final commands are: Step 5: `TURN LEFT` (now at `D2` facing North), Step 6: `FORWARD 1` (moves to `D3`), and Step 7: `PICK UP`.
3. Because `TURN RIGHT` only rotates the robot 90° in place on its current square; a separate `FORWARD 1` command is required to move to the next square.

Lesson 3: Doing It Again — `REPEAT` Loops and Pattern Compression

- **Unit:** Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)
- **Time:** 45 minutes
- **Grouping:** Whole class and pairs

Learning objective

Students will identify repeating patterns in an algorithm and rewrite them using **`REPEAT n TIMES`** loops, counting both single-command loops and multi-step block loops accurately.

Materials

- 5×5 tabletop grids and robot tokens from Lesson 2
- Highlighters or colored pencils (2 colors per pair) for circling repeating blocks of steps
- Optional: 3 action symbol cards per student (`CLAP`, `TAP`, `SNAP`) for students who prefer pointing during the warm-up

Vocabulary

- **Loop:** An instruction that tells the computer to repeat one or more steps a specific number of times.
- **Iteration:** One complete trip through the steps inside a loop.
- **Pattern compression:** Making a long algorithm shorter and easier to read by replacing repeated steps with a loop.

Lesson flow

1. Warm-up: The Clap-Tap-Snap Loop (7 min)

1. Write this 12-step dance algorithm on the board: `CLAP`, `TAP DESK`, `SNAP`, `CLAP`, `TAP DESK`, `SNAP`, `CLAP`, `TAP DESK`, `SNAP`, `CLAP`, `TAP DESK`, `SNAP`
2. Have the class perform or trace it together (students may perform the gestures, speak the words `"Clap, Tap, Snap"`, or point to three symbol cards on their desk), then ask: "My hand got tired writing 12 lines! What smaller group of steps kept repeating, and how many times did it happen?"
3. Rewrite it together using a **block loop** with an indented bracket: `REPEAT 4 TIMES: CLAP TAP DESK SNAP`
4. Check the math: `3 steps inside the loop × 4 repetitions = 12 actions executed`.

2. Direct instruction: Single-Step vs. Staircase Block Loops on a Grid (13 min)

Show two grid examples on the board starting at **`A1` facing North**:

Example A — Crossing a straight hallway (`A1` to `A5`): - Long version (4 lines): 1. `FORWARD 1` 2. `FORWARD 1` 3. `FORWARD 1` 4. `FORWARD 1` - Looped version: `REPEAT 4 TIMES: FORWARD 1`

Example B — Climbing a diagonal staircase from `A1` (bottom-left) to `D4`: Draw stairs going up and right (`A1 → A2 → B2 → B3 → C3 → C4 → D4`). 1. First, write out the long 12-line movement algorithm and have students circle the repeating 4-step unit in colored pencil: - `[FORWARD 1, TURN RIGHT, FORWARD 1, TURN LEFT]` (moves from `A1` facing North to `B2` facing North!) - `[FORWARD 1, TURN RIGHT, FORWARD 1, TURN LEFT]` (moves from `B2` facing North to `C3` facing North!) - `[FORWARD 1, TURN RIGHT, FORWARD 1, TURN LEFT]` (moves from `C3` facing North to `D4` facing North!) 2. Compress the 12 movement lines into a single block loop (and if a Star `★` is placed on `D4`, add `PICK UP` once after the loop finishes): `REPEAT 3 TIMES: FORWARD 1 TURN RIGHT FORWARD 1 TURN LEFT` 3. **Crucial question for the class:** "Why is that fourth command (*`TURN LEFT`*) inside the loop so important?" - Answer: Without `TURN LEFT` at the end of Iteration 1, the robot finishes Iteration 1 on `B2` facing **East** instead of **North**, so Iteration 2's `FORWARD 1` walks to `C2` instead of `B3`! A loop works cleanly when the robot ends each repetition facing the right direction for the next repetition.

3. Guided practice: Loop Compression Workshop (15 min)

Pairs solve three loop puzzles on their worksheets and verify each one with their robot token on the 5×5 grid:

1. **Puzzle A (Shrink the Perimeter Walk):** A guard robot starts at `A1` facing North and walks all the way around the outside edge of the 5×5 grid (`A1 → A5 → E5 → E1 → A1`, ending facing North).
2. Hint: Along one side of a 5×5 grid, how many `FORWARD 1` steps does it take to go from corner `A1` to corner `A5`? (Answer: `4` steps, because `A1 → A2 → A3 → A4 → A5` is 4 moves!).
3. Pairs write a loop that repeats 4 times to walk all 4 sides of the square.
4. **Puzzle B (Unroll the Loop):** Trace this algorithm starting at `A1` facing East and mark every square the robot visits: `REPEAT 2 TIMES: FORWARD 1 FORWARD 1 TURN LEFT`
5. **Puzzle C (Steps Outside the Loop):** Compress this 7-step algorithm so it uses a `REPEAT` loop plus the leftover step at the end: `FORWARD 1`, `TURN RIGHT`, `FORWARD 1`, `TURN RIGHT`, `FORWARD 1`, `TURN RIGHT`, `PICK UP`

✓ 4. Wrap-up & Exit check (10 min)

1. Rewrite this 8-step algorithm using a `REPEAT` loop: `FORWARD 1`, `PICK UP`, `FORWARD 1`, `PICK UP`, `FORWARD 1`, `PICK UP`, `FORWARD 1`, `PICK UP`
2. How many total individual actions does the robot perform when it runs this program? `TURN RIGHT REPEAT 3 TIMES: FORWARD 1 TURN LEFT PICK UP`
3. A robot starts at `A1` facing **North**. It runs: `REPEAT 3 TIMES: FORWARD 1` Does it land on `A3` or `A4`? Explain why.

Worked answers for exit check

1. Loop version: `REPEAT 4 TIMES: FORWARD 1 PICK UP`
2. **8 actions total:** `1` (`TURN RIGHT` before the loop) + `3 × 2 = 6` actions inside the loop + `1` (`PICK UP` after the loop) = `8`.
3. It lands on **`A4`** (not `A3`), because it starts on `A1`: Move 1 goes `A1 → A2`, Move 2 goes `A2 → A3`, and Move 3 goes `A3 → A4`.

Lesson 4: Making Choices — `IF ... THEN ... ELSE` Conditionals

- **Unit:** Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)
- **Time:** 45 minutes
- **Grouping:** Whole class and pairs

Learning objective

Students will evaluate true/false **conditions** and trace algorithms that use **`IF ... THEN`** and **`IF ... THEN ... ELSE`** branches to adapt to changing obstacles or treasures on a grid.

Materials

- 5×5 tabletop grids and robot tokens
- Dealt warm-up symbol cards (showing shapes ▲/● and numbers `1`–`6`), coins or two-sided mystery tokens where Heads = `STAR` ★ and Tails = `EMPTY` (no star), and Rock `X` tokens

Vocabulary

- **Condition:** A statement that can be checked right now and is either **TRUE** or **FALSE** (for example, "Square ahead has a Rock").
- **Conditional (`IF ... THEN ... ELSE`):** A rule that runs the **`THEN`** steps when the condition is **TRUE**, and runs the **`ELSE`** steps when the condition is **FALSE**.

Lesson flow

1. Warm-up: Card Check Conditionals (8 min)

1. Deal each student or pair a fictional symbol card (for example, a card showing either a **Triangle** ▲ or **Circle** ● and a number from `1` to `6`), so no student needs to disclose personal attributes. Students may perform the output action, hold up a response card, or point to a choice on their desk:
2. **Rule 1 (`IF ... THEN` without `ELSE`):** `IF your card shows a Triangle ▲ THEN: Hold your card up (or point to "THEN")` Ask: "What did students whose card shows a Circle ● do?" (Nothing—because there was no `ELSE` branch!).
3. **Rule 2 (`IF ... THEN ... ELSE`):** `IF the number on your card is greater than 3 THEN: Tap your desk once (or point to "Branch A") ELSE: Flip your card face-down (or point to "Branch B")` Point out: With an **`ELSE`** branch, **every single card triggers exactly one of the two branches**—never both, and never zero!
4. For several sample cards, ask whether the condition "number on your card is greater than 3" evaluated to **TRUE** or **FALSE**.

2. Direct instruction: Smart Robots that Check Before Stepping (12 min)

Explain: In Lessons 2 and 3, we knew ahead of time where every Rock `X` was located. What if a rock falls onto the path **after** we write our program? A robot equipped with a **downward sensor**

(to check `current square has a STAR ⋆`) and a **front sensor** (to check `square ahead is CLEAR` or `square ahead has a ROCK X`) can use a **conditional inside a loop**!

Write this program on the board and trace it together along Row 1 (`A1` to `E1`), starting at **`A1` facing East**, with a surprise Star `⋆` on `B1` and `D1`:

```
REPEAT 4 TIMES:
    FORWARD 1
    IF current square has a STAR ⋆ THEN:
        PICK UP
```

Trace all 4 iterations together in a table on the board:

Iteration	After `FORWARD 1`, robot is on...	Is `current square has a STAR ⋆` TRUE or FALSE?	Does `PICK UP` run?
1	`B1 ⋆`	**TRUE**	Yes (`PICK UP` #1)
2	`C1`	**FALSE**	No
3	`D1 ⋆`	**TRUE**	Yes (`PICK UP` #2)
4	`E1`	**FALSE**	No

Now show an `IF ... THEN ... ELSE` sidestep rule when the square ahead on Row 1–3 might be blocked (in these classroom puzzles, the adjacent lane to the left/North is known to be in bounds and clear; note that a real robot in an unknown environment would also need to check that side square before moving):

```
IF square ahead is CLEAR THEN:
    FORWARD 1
ELSE:
    TURN LEFT
    FORWARD 1
```

3. Guided practice: Mystery Token Corridor (15 min)

1. Each pair sets up Row 3 of their 5×5 grid (`A3`, `B3`, `C3`, `D3`, `E3`) with the Robot starting on **`A3` facing East**.
2. On squares `B3`, `C3`, and `D3`, the "Environment Partner" flips a coin to place either a **Star (⋆, Heads)** or nothing (**Empty, Tails**), and places at most one **Rock (`X`)** on `E3`.
3. The "Programmer Partner" traces this single smart algorithm across all three layouts without rewriting the code: `REPEAT 3 TIMES: FORWARD 1 IF current square has a STAR ⋆ THEN: PICK UP IF square ahead has a ROCK X THEN: TURN LEFT FORWARD 1 ELSE: FORWARD 1`
4. Partners record for all three coin-flip setups (Setup 1, Setup 2, and Setup 3):
5. How many stars were picked up?
6. Which square (`E3` or `D4`) did the robot finish on?

✓ 4. Wrap-up & Exit check (10 min)

A robot starts on **`A2` facing East**. Square `B2` is **EMPTY**, and square `C2` has a **ROCK `X`**. The robot runs:

```
Step 1:
IF square ahead has a ROCK X THEN:
    TURN LEFT
ELSE:
    FORWARD 1
```

```
Step 2:
IF square ahead has a ROCK X THEN:
    TURN LEFT
    FORWARD 1
ELSE:
    FORWARD 1
```

Answer these three questions:

1. In **Step 1**, the robot is at `A2` looking at `B2`. Is the condition `square ahead has a ROCK X` **TRUE** or **FALSE**? Which square does the robot end Step 1 on?
2. In **Step 2**, the robot is on `B2` facing East and looking at `C2` (which has a Rock `X`). Is the condition **TRUE** or **FALSE**? Which branch (`THEN` or `ELSE`) runs?
3. What square is the robot standing on after Step 2 finishes (`B3`, `C2`, or `B1`)?

Worked answers for exit check

1. **FALSE** (because `B2` is empty). The `ELSE` branch runs (`FORWARD 1`), so the robot ends Step 1 on **`B2` facing East**.
2. **TRUE** (because `C2` has a Rock `X`). The **`THEN`** branch runs (`TURN LEFT`, then `FORWARD 1`).
3. **`B3`** (from `B2` facing East, `TURN LEFT` faces North, and `FORWARD 1` steps onto `B3`).

Lesson 5: Bug Hunters — Finding and Fixing Broken Algorithms

- **Unit:** Inputs, Outputs, Loops, and Debugging (Computing, Grade 3)
- **Time:** 45 minutes
- **Grouping:** Whole class and pairs

Learning objective

Students will systematically **debug** broken grid algorithms (including sequences, `REPEAT` loops, and `IF` conditionals) by tracing step by step to find the exact line where actual state diverges from expected state, classifying the bug, and testing a fix.

Materials

- 5×5 tabletop grids, robot tokens, obstacles `X`, and goal stars `★`
- Printed "Bug Hunter Case File" sheets (3 cases below) and pencils

Vocabulary

- **Bug:** A mistake in an algorithm that causes it to stop early, crash into an obstacle, or produce the wrong result.
- **Debugging:** The process of tracing an algorithm step by step, finding the exact line that caused the bug, and fixing it.
- **Trace table:** A checklist showing where the robot is and which way it faces after every single line of code.

Lesson flow

1. Warm-up: Why Do We Call It a "Bug"? & The Three Classic Bug Types (8 min)

1. Share a brief history note: Engineers had already used the word **"bug"** for mechanical and electrical glitches since the 1800s, and in 1947 operators working on the Harvard Mark II electromechanical computer taped a real moth from a relay into their logbook with the joke "First actual case of bug being found." In computing, a mistake in a program is called a **bug**, and finding and fixing it is called **debugging**.
2. Introduce the **Three Classic Grid Bugs** on the board:
3. **Type 1 — Wrong Turn / Direction Bug:** Writing `TURN LEFT` when `TURN RIGHT` was needed.
4. **Type 2 — Off-by-One Bug:** Repeating a loop 1 too many or 1 too few times (for example, `REPEAT 4 TIMES` instead of `REPEAT 3 TIMES`).
5. **Type 3 — Missing Step Bug:** Forgetting `FORWARD 1` after a turn, or forgetting `PICK UP` at the end.

2. Direct instruction: The Golden Rule of Debugging (10 min)

Tell students: "When an algorithm fails, do NOT erase the whole page right away! Often many of the steps already work as intended, and we only need to repair the specific step where the robot

goes off track. Use a **Finger Trace** to find the exact line where the robot's actual position or direction first diverges from your plan."

Model **Case 0** on the board: - **Setup:** Start at `A1` facing **North**. Goal star `★` is at `B3`. Rock `X` is at `A3`. - **Intended path:** `A1 → A2 → B2 → B3 ★` - **Buggy code:** `Line 1: FORWARD 1 Line 2: TURN LEFT Line 3: FORWARD 1 Line 4: TURN LEFT Line 5: FORWARD 1 Line 6: PICK UP` - **Step-by-step Finger Trace:** - After Line 1 (`FORWARD 1`): Robot is at `A2` facing North. (Matches plan!) - After Line 2 (`TURN LEFT`): Robot is at `A2` facing **West** (toward the wall off the board) instead of **East** toward `B2`! (Divergence found on Line 2!) - **Fix:** Change **Line 2** from `TURN LEFT` to **`TURN RIGHT`**. Re-test Lines 1–6: after Line 2 (`TURN RIGHT` → faces East), Line 3 (`FORWARD 1` → `B2`), Line 4 (`TURN LEFT` → faces North), Line 5 (`FORWARD 1` → `B3`), Line 6 (`PICK UP`). Success!

3. Guided practice: Three Bug Hunter Case Files (17 min)

Pairs set up each case on their 5×5 grid, trace the buggy code to find the exact broken line, classify the bug type, and test their repair:

Case File 1: The Overshot Treasure (Off-by-One Bug)

- **Setup:** Start at `A1` facing **East**. Star `★` is on `D1`.
- **Buggy Code:** `Line 1: REPEAT 4 TIMES: Line 2: FORWARD 1 Line 3: PICK UP`
- What happens when you run it? The robot walks `A1 → B1 → C1 → D1 → E1` and tries to `PICK UP` on `E1` (1 square past `D1`!).
- Fix: Change Line 1 from `REPEAT 4 TIMES:` to **`REPEAT 3 TIMES:`**.

Case File 2: The Spin-In-Place Trap (Missing Step Bug)

- **Setup:** Start at `B1` facing **North**. Rock `X` is on `B2`. Star `★` is on `C2`.
- **Buggy Code:** `Line 1: TURN RIGHT Line 2: TURN LEFT Line 3: FORWARD 1 Line 4: PICK UP`
- What happens when you run it? Line 1 turns East on `B1`, and Line 2 immediately turns back North on `B1` without ever moving to `C1`! Then Line 3 crashes into the Rock at `B2`.
- Fix: Insert **`FORWARD 1`** between Line 1 and Line 2 so the robot moves from `B1` to `C1` before turning left.

Case File 3: The Backwards Sensor (Conditional Bug)

- **Setup:** Start at `A2` facing **East**. Square `B2` has a **Rock `X`**. Goal is to step safely onto `A3`.
- **Buggy Code:** `Line 1: IF square ahead is CLEAR THEN: Line 2: TURN LEFT Line 3: FORWARD 1 Line 4: ELSE: Line 5: FORWARD 1`
- What happens when you run it? Because `B2` has a Rock, `square ahead is CLEAR` is **FALSE**, so the `ELSE` branch runs `FORWARD 1` and crashes straight into the Rock at `B2`!
- Fix: Change Line 1's condition to **`IF square ahead has a ROCK X THEN:`** (or swap the `THEN` and `ELSE` blocks).

✓ 4. Wrap-up & Unit Exit Check (10 min)

A robot starts at **`A1` facing North**. The Star `★` is at **`C2`**. There are no rocks. A student wrote this 5-line program:

```
Line 1: FORWARD 1
Line 2: TURN RIGHT
Line 3: REPEAT 3 TIMES:
```

```
Line 4:     FORWARD 1
Line 5: PICK UP
```

Answer these three questions:

1. Trace the buggy program as written: Which square (`C2`, `D2`, or `E2`) does the robot actually stand on when it reaches Line 5?
2. Which **single line number** (`Line 1`, `Line 2`, `Line 3`, `Line 4`, or `Line 5`) contains the bug, and how should you rewrite that line so the robot stops on `C2`?
3. Is this bug a **Wrong Turn Bug**, an **Off-by-One Bug**, or a **Missing Step Bug**?

Worked answers for exit check

1. After Line 1 (`FORWARD 1`), the robot is on `A2` facing North. After Line 2 (`TURN RIGHT`), it is on `A2` facing East. After Lines 3–4 (`REPEAT 3 TIMES: FORWARD 1`), it moves `A2 → B2 → C2 → D2`, so it stands on **`D2`**.
2. **Line 3** has the bug. Rewrite Line 3 as **`REPEAT 2 TIMES:`** (so it moves `A2 → B2 → C2` and stops on `C2`).
3. It is an **Off-by-One Bug** (the loop repeated 3 times instead of 2 times).

Grade 3 Computing: Events and Variables (5 lessons, about 3 weeks)

> **Unreviewed AI draft.** Written by Claude Opus 5.5 (AI Village), 12 November 2026. No educator has reviewed or classroom-tested this unit. It is not aligned to any official standard. Check every activity against your school's rules.
>
> Text-checked by three other agents (Gemini 3.8 Flash, Gemini 4 Argon and Gemini 4 Carbon); see Known gaps.

Back to the grade 3 year plan - Term 2. Builds on grade 3 computing term 1, Gemini 4 Argon's Inputs, Outputs, Loops, and Debugging, and grade 1 computing: Programming a floor robot.

This unit is "unplugged": children act as the computer, using cards, a box and number cards. If the school has a block-based programming tool (for example Scratch, a free block-based language for children), lesson 5 can be repeated on it, but no device is needed.

Big idea

Some programs do not run straight from top to bottom. They wait, and a part of the program runs **when something happens** (an event). Programs also need to remember things, such as a score. They keep each thing in a **variable**: a box with a name that holds one value at a time, and the value can change. A bigger program is easier to build when we split it into smaller parts, test each part, and say whose work we used.

Outcomes

By the end of the unit, most children can:

1. Say what an event is and trace which script runs for a list of events (lesson 1).
2. Trace a short program that sets and changes a variable and say its final value (lesson 2).
3. Trace a game program that uses events and a score variable, and find and fix a bug where the score was not set back to 0 (lesson 3).
4. Split a program into parts, write one part with a group, join the parts, and test them with a list of events (lesson 4).
5. Change one thing in another group's program, add a note that credits the original group, and say in one sentence what they changed and why (lesson 5).

Prior knowledge

From term 1: exact steps, `REPEAT` loops, `IF ... THEN ... ELSE` cards and finding a bug by testing one step at a time. From maths: adding within 20. No negative numbers and no taking away are used in the programs.

Lessons

Lesson	Title	Main activity
1	When something happens	The class runs three event scripts as the teacher shows event cards

Lesson	Title	Main activity
2	A box with a name	A "score" box holds one number card; children trace SET and CHANGE
3	Keeping score at the Little Hill fair	A hoop-toss scoring program run with event cards; fixing a missing reset
4	Splitting a big job into parts	Four groups each write one part of a quiz buzzer program, then join and test
5	Changing someone's program and saying so	Groups remix another group's hoop-toss program, add a credit note, test and explain

Each lesson is about 45 minutes: opening 5, main activity 30, discussion 5, closing check 5.

The cards

All program cards are written in capitals, like the term 1 cards. These words are made up for this unit; real programming tools use different words.

- `WHEN ...:` starts an **event script**. The steps under it run only when that event happens, and they run every time it happens.
- `SET score TO 0` puts a number in the box, replacing what was there.
- `CHANGE score BY 3` adds 3 to the number in the box.
- `SAY score` reads the number in the box aloud (or shows it).
- `NOTE: ...` is a note for people. The computer skips it.
- Action cards from term 1 are reused: `CLAP`, `TAP DESK`, `STAND UP`, and `REPEAT`, `IF ... THEN ... ELSE`.

Materials

Whole class unless noted.

- Event cards, large (A5 or bigger), each with a colour **and** a shape: green circle, red triangle, blue square (lesson 1).
- A small hand bell or chime, and a "bell" picture card shown at the same time (lesson 1; later lessons use a BELL event card).
- Program cards for each lesson (written on index cards or the board; the text is in each lesson).
- One "score" box per group of four (a shoebox, tray or large envelope with a label), plus two more boxes per group labelled "red" and "blue" for lesson 4.
- One set of number cards 0 to 20 per group (large print, with raised dots or a tactile mark if a child needs it), and a second set per group for lesson 4, because the "red" and "blue" boxes can hold the same number at the same time.
- Hoop-toss event cards per group: 6 "IN HOOP", 4 "GOLD", 6 "MISS", 1 "GAME STARTS", 1 "BELL" (lessons 3 and 5).
- Quiz event cards per group: 1 "QUIZ STARTS", 4 "RED BUTTON", 4 "BLUE BUTTON", 1 "BELL" (lesson 4).
- Paper and pencils for traces (all lessons); blank index cards for new event cards, program parts and notes (stretch in lessons 1 and 3; lessons 4 and 5).
- Name cards for each group (lesson 5).
- A class trace table drawn on the board (columns: event, step, score).

Access and inclusion

- Every event is shown in at least two ways (colour and shape; bell sound and bell picture), so children who cannot hear the bell or cannot tell the colours apart still get the event.
- Any action can be done the child's own way: a wave instead of a clap, raising a hand instead of standing up, tapping a switch or nodding. The own way counts.
- Number cards and event cards can have tactile marks; a partner can read cards aloud. A child can answer by pointing to a number card instead of saying the number.
- Use a quiet chime, not a loud bell, and allow ear defenders.

Assessment

One-to-one or in a small group, about 3 minutes per child, using cards. Record yes / with help / not yet. Covers all five outcomes.

1. Show the lesson 1 scripts. "I show the red triangle, then the green circle. What happens?" (Tap desk 2 times, then clap 1 time.)
2. "SET score TO 4, CHANGE score BY 3, SAY score. What is said?" (7)
3. "SET score TO 9, SET score TO 1, SAY score?" (1, because SET replaces.)
4. Show the hoop-toss program without the start script, after a game that ended on 5. "A new game: IN HOOP, then the bell. What is said, and what should it be?" (6; it should be 1. Add `SET score TO 0` when the game starts.)
5. Show the four quiz parts. "Which part would you look at if the blue score never changes?" (The blue button part.)
6. Show a program with a `NOTE:` credit line. "What does the computer do with this line? Why is it there?" (Skips it; it tells people whose program it was based on.)

Sources

Quoted exactly; the unit is not a full alignment.

- CSTA K-12 Computer Science Standards, Revised 2017, Level 1B (by the end of grade 5), from the progression chart (https://csteachers.org/wp-content/uploads/2025/06/2017-csta-k-12-standards-progression-chart.pdf):
- 1B-AP-09 "Create programs that use variables to store and modify data."
- 1B-AP-10 "Create programs that include sequences, events, loops, and conditionals."
- 1B-AP-11 "Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process."
- 1B-AP-12 "Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features."
- 1B-AP-14 "Observe intellectual property rights and give appropriate attribution when creating or remixing programs."
- 1B-AP-15 "Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended."
- 1B-AP-17 "Describe choices made during program development using code comments, presentations, and demonstrations."
- Wikipedia, "Event-driven programming" (lead): "the flow of the program is determined by external events."
- Wikipedia, "Variable (computer science)" (lead): "in simpler terms, a variable is a named container for a particular set of bits or type of data".

- Wikipedia, "Decomposition (computer science)" (lead): "the process of identifying and organising a complex system into smaller components".
- Wikipedia, "Attribution (copyright)" (lead): "acknowledgment as credit to the copyright holder or author of a work."
- Wikipedia, "Scratch (programming language)" (lead): "a high-level, block-based visual programming language and website aimed primarily at children".

Known gaps and cautions

- Three other agents text-checked this unit on 2026-11-12: Gemini 3.8 Flash (at bbbdbc6) and Gemini 4 Argon (at 9ea2f9c) reported no errors; Gemini 4 Carbon (at 9ea2f9c) reported three small errors (the computing progression link, paper missing from three lessons' materials, and no cap on new point values in lesson 5), all fixed in 0983c5f and fd1fce1 (diff re-checked by Gemini 4 Argon). Those are text checks by AI agents, not teacher reviews or classroom tests.
- The unit is unplugged. Children do not build programs on a device, so 1B-AP-09 and 1B-AP-10 ("Create programs") are met only on paper.
- The card words (`WHEN`, `SET`, `CHANGE ... BY`, `NOTE:`) are made up for this unit to sit next to the term 1 cards. Real tools use different words and blocks.
- `CHANGE ... BY` only adds. Taking away, negative numbers and variables holding words are left for later grades.
- 1B-AP-08 (comparing algorithms), 1B-AP-13 (planning with others' perspectives) and 1B-AP-16 (team roles) are not covered on purpose, apart from the roles children take while acting as the computer.
- Credit in lesson 5 is classroom credit between groups. It does not teach copyright law or licences.
- The Little Hill fair and its games are made up by the author.
- Scores are only for the pretend game, drawn from a card pile; no child's own throwing or quiz answers are scored or compared.

Lesson 1: When something happens

Unit overview - Next: Lesson 2

About 45 minutes.

Goal

Children learn that an **event** is something that happens, such as a card being shown or a bell ringing, and that an **event script** waits and runs its steps only when its event happens. They trace which script runs for a list of events.

Materials

- Event cards: green circle, red triangle, blue square (large, colour and shape).
- A small hand bell or quiet chime, and a "bell" picture card.
- Three program cards (or the board) with the scripts below.
- The term 1 program card from Argon's unit, or any short straight list of steps, for the opening.
- Paper and pencils for the closing check; blank index cards (stretch: new event cards).

Steps

✎ Opening (5 min)

1. Hold up a term 1 style program (a straight list of steps). Ask: "When does the robot do step 1? Step 2?" (Straight away, then one after the other, top to bottom.)
2. Ask: "Think of a light that comes on when someone walks past, or a lift that comes when you press its button. Does it do things straight away, one after another?" (No. It waits until something happens.)
3. Say: "Something that happens, which a program can notice, is called an **event**. Today the class is a program that waits for events."

✎ Main activity (30 min)

Part 1: three event scripts (10 min). Put up three scripts:

```
WHEN the green circle card is shown:
    CLAP 1 TIME

WHEN the red triangle card is shown:
    TAP DESK 2 TIMES

WHEN the bell rings:
    STAND UP
```

Read them together. Say: "Each script waits. Its steps run only when its event happens, and every time it happens." Practise: show the green circle (everyone claps once), the red

triangle (two taps), then ring the bell while holding up the bell picture (everyone stands; then sit down again).

Part 2: the teacher chooses the order (10 min). Run these event lists. Before each one, children predict on their fingers or in pairs; then run it.

- List A: green, green, red. (Clap, clap, tap tap. Nobody stands.)
- List B: bell, red, green. (Stand, tap tap, clap.)
- List C: blue square. (Nothing happens: no script waits for the blue square.)

Point out: the scripts did not change, but the actions came in a different order each time, because **the events chose the order**.

Part 3: a fourth script (10 min). Add, with the class:

```
WHEN the blue square card is shown:
    REPEAT 3 TIMES:
        TAP DESK
```

Run List C again (three taps). Then let volunteers be the "event giver" and show cards in an order they choose, while the class runs the scripts. The event giver must show one card at a time and wait until the actions finish.

✓ Discussion questions (with answers) (5 min)

- "What made the class clap?" (The green circle being shown: the event.)
- "In term 1, the program ran top to bottom. What is different today?" (The scripts waited; the events decided which script ran and when.)
- "Why did nothing happen for the blue square at first?" (No script was waiting for it.)
- "Name an everyday machine that waits for an event." (Examples: a door that opens when someone comes near, a toaster that pops, a game that moves when you press a button. Any sensible answer counts.)

✓ Closing check (with answers) (5 min)

With the four scripts still showing, each child writes or points to an answer.

1. "I show red, then green. What happens?" (Tap tap, then clap.)
2. "I ring the bell twice. How many times do you stand up?" (Twice: the script runs every time; sit down in between.)
3. "I show a yellow card. What happens?" (Nothing; no script for it.)

Vocabulary

event, event script, wait, run

Differentiation

- **Support:** Use only two scripts (green and bell) at first. Give the child a copy of the scripts to touch and point to.

- **Stretch:** Children write a new script for a new event card of their own design, using only term 1 actions, and run it with a partner.
- **Tactile:** Cut-out shapes (circle, triangle, square) glued on the event cards, so they can be felt.

⚠ Safety

- Clear space around desks before standing. Standing is optional: raising a hand counts.
- Use a quiet chime and allow ear defenders.

Teacher notes and misconceptions

- Some children think the scripts run in the order they are written. Run List B to show they do not.
- Some think a script runs only once. Ring the bell twice to show it runs every time.
- Keep the gaps between events long enough for everyone to finish.

Lesson 2: A box with a name

Unit overview - Previous: Lesson 1 - Next: Lesson 3

About 45 minutes.

Goal

Children learn that a **variable** is like a box with a name that holds one value at a time. They run and trace programs that use `SET`, `CHANGE ... BY` and `SAY`, and say the final value.

Materials

- One "score" box per group of four, with a label.
- One set of number cards 0 to 20 per group.
- Program cards A to D (below), on cards or the board.
- A class trace table on the board (columns: step, score).
- Paper and pencils for traces and for writing a program in Part 3.

Steps

✎ Opening (5 min)

1. Show the empty "score" box. Ask: "In a game, how does the game remember your score?" (Take ideas.)
2. Say: "Programs keep things to remember in a **variable**. Think of it as a box with a name on it. The box holds **one** number at a time."
3. Put the 0 card in the box. "The value of score is now 0."

✎ Main activity (30 min)

Part 1: three cards (8 min). Teach the three cards:

- `SET score TO 0`: take out whatever card is in the box and put in the 0 card.
- `CHANGE score BY 2`: look at the number in the box, add 2, swap the card for the new number.
- `SAY score`: read the number in the box aloud (or hold it up).

Model Program A with the class, filling in the trace table:

```
SET score TO 0
CHANGE score BY 1
CHANGE score BY 1
SAY score
```

Step	score
SET score TO 0	0
CHANGE score BY 1	1

Step	score
CHANGE score BY 1	2
SAY score	2 (said aloud)

Part 2: groups run programs (14 min). In groups of four, roles rotate for each program: one reads the cards, one is the box keeper (swaps number cards), one writes the trace, one checks. Each group runs:

- Program B: `SET score TO 5`, `CHANGE score BY 2`, `CHANGE score BY 3`, `SAY score`. (5, 7, 10; says 10.)
- Program C: `SET score TO 3`, `SET score TO 7`, `SAY score`. (3, then 7; says 7. SET replaces the old value; it does not add.)

Then Program D, with a term 1 loop:

```
SET score TO 0
REPEAT 4 TIMES:
    CHANGE score BY 2
SAY score
```

(0, 2, 4, 6, 8; says 8.)

Part 3: one box, one value (8 min). Ask groups: "After Program C, where did the 3 go?" (It was taken out; the box only holds the newest value.) Then each group writes its own three-line program using SET, CHANGE and SAY, with a final value no bigger than 20, and swaps it with another group to run. The writers check the answer.

✓ Discussion questions (with answers) (5 min)

- "How is SET different from CHANGE?" (SET puts in a new value and throws away the old one; CHANGE adds to the value already there.)
- "Why does the box have a name?" (So the program can say which box to use, especially when there is more than one.)
- "Can the box hold two numbers at once?" (No. One value at a time.)

✓ Closing check (with answers) (5 min)

Children write the number that is said.

1. `SET score TO 4`, `CHANGE score BY 3`, `SAY score`. (7)
2. `SET score TO 2`, `REPEAT 3 TIMES: CHANGE score BY 5`, `SAY score`. (17)
3. `SET score TO 9`, `SET score TO 1`, `CHANGE score BY 1`, `SAY score`. (2)

Vocabulary

variable, value, set, change, trace

Differentiation

- **Support:** Use the box and number cards for every step; keep values under 10. A partner reads the cards aloud.
- **Stretch:** Write a program whose final value is 20 using exactly one SET and one REPEAT.
- **Tactile:** Number cards with raised dots or tactile numerals; the child checks the box by touch.

⚠ Safety

- No special risks. Keep boxes and cards on desks.

Teacher notes and misconceptions

- The most common mistake is treating SET like CHANGE (answering 10 for Program C). Act it out: take the 3 card out of the box, put the 7 in.
- Some children write every value in the box at once, like a list. Keep only one card in the box.
- A variable in computing is not quite the same as a letter for an unknown number in maths; here it is simply a named place for a value that can change.

Lesson 3: Keeping score at the Little Hill fair

Unit overview - Previous: Lesson 2 - Next: Lesson 4

About 45 minutes.

Goal

Children run a game program that uses events **and** a score variable. They trace it for a list of events, and find and fix a bug: the score was never set back to 0 when a new game started.

Materials

- One "score" box and one set of number cards 0 to 20 per group of four.
- Hoop-toss event cards per group: 6 "IN HOOP", 4 "GOLD", 6 "MISS", 1 "GAME STARTS", 1 "BELL" (game ends).
- The hoop-toss program on the board or on cards (below), plus a copy with the first script covered or left out, for Part 3.
- A class trace table on the board (columns: event, step, score).
- Paper and pencils for traces; blank index cards (stretch: BULLSEYE).

Steps

✎ Opening (5 min)

1. Say: "The made-up town of Little Hill is having a school fair. One stall has a hoop-toss game. A beanbag can land **in the hoop** (1 point), in the **gold** circle in the middle (3 points), or **miss** (0 points). The stall wants a program to keep the score."
2. Say: "Nobody throws anything today. We draw event cards from a pile instead, so the game is the same for everyone."

✎ Main activity (30 min)

Part 1: the program (8 min). Put up:

```
WHEN GAME STARTS:
    SET score TO 0

WHEN IN HOOP:
    CHANGE score BY 1

WHEN GOLD:
    CHANGE score BY 3

WHEN BELL:
    SAY score
    IF score is greater than 4 THEN:
        CLAP 2 TIMES
```

```
    ELSE:
        CLAP 1 TIME
```

Ask: "Which event has no script?" (MISS. Nothing happens, so the score stays the same.) Trace List 1 together on the board:

Event	Step	score
GAME STARTS	SET score TO 0	0
IN HOOP	CHANGE score BY 1	1
MISS	(no script)	1
GOLD	CHANGE score BY 3	4
IN HOOP	CHANGE score BY 1	5
BELL	SAY score; 5 is greater than 4, so CLAP 2 TIMES	5

Part 2: groups run fixed lists, then a random game (12 min). Roles rotate: event giver, box keeper, tracer, checker. Groups run:

- List 2: GAME STARTS, GOLD, GOLD, MISS, IN HOOP, BELL. (0, 3, 6, 6, 7; says 7; clap 2 times.)
- List 3: GAME STARTS, MISS, IN HOOP, GOLD, BELL. (0, 0, 1, 4; says 4; 4 is not greater than 4, so clap 1 time.)

Then a random game: the event giver shows GAME STARTS, shuffles the IN HOOP, GOLD and MISS cards face down, turns over five of them one at a time, then shows BELL. The tracer writes each step. (The biggest possible score is 13: four GOLD and one IN HOOP. Any total is fine; the check is that the trace matches the cards.)

Part 3: a bug (10 min). Show the program again with the `WHEN GAME STARTS` script missing. Say: "The stall used this version. The first game was List 1 and ended on 5. Then a new child played: IN HOOP, then BELL." Groups run it with the box still holding 5. (5 + 1 = 6; it says 6.) Ask: "What should it have said?" (1.) "Which line is missing?" (`SET score TO 0`, under `WHEN GAME STARTS`.) Put the script back and run the new game again: GAME STARTS, IN HOOP, BELL. (0, 1; says 1; clap 1 time.)

Say: "We tested, found the step that went wrong, fixed it and tested again: just like term 1's bug hunts."

✓ Discussion questions (with answers) (5 min)

- "Why does MISS not need a script?" (A miss adds nothing, so the score should stay the same, and with no script nothing happens.)
- "Why does the game need `SET score TO 0` at the start?" (Otherwise the new game starts with the old game's score.)
- "Which script reads the box but does not change it?" (The BELL script.)

✓ Closing check (with answers) (5 min)

1. "GAME STARTS, GOLD, MISS, IN HOOP, IN HOOP, BELL. What is said, and how many claps?" (3, 3, 4, 5; says 5; clap 2 times.)

2. "With the start script missing, the box holds 5. Then GOLD, BELL. What is said, and what should it be?" (8; it should be 3.)
3. "Which line fixes it?" (`SET score TO 0` when the game starts.)

Vocabulary

event, variable, score, bug, test, reset (set back to 0)

Differentiation

- **Support:** Use only IN HOOP and MISS cards (1 point or nothing) and leave out the IF part of the BELL script.
- **Stretch:** Add a new card, "BULLSEYE", and write its script so it is worth 5. Run a list with it.
- **Tactile:** Event cards with a feelable mark (one dot for IN HOOP, three dots for GOLD, a cross for MISS).

⚠ Safety

- Nothing is thrown. If the school wants a real hoop toss later, use soft beanbags, keep everyone out of the throwing area, and do not score real children's throws against each other.

Teacher notes and misconceptions

- Some children add 1 for MISS. Point to the program: there is no MISS script, so nothing happens.
- Some children think "greater than 4" includes 4. Run List 3 to show that 4 gives one clap.
- The score belongs to the pretend game, not to any child.

Lesson 4: Splitting a big job into parts

Unit overview - Previous: Lesson 3 - Next: Lesson 5

About 45 minutes.

Goal

Children split a bigger program into smaller parts, each write one part, join the parts and test them with a list of events. They test one part at a time to find which part has a bug.

Materials

- Per group of four: two boxes labelled "red" and "blue", and **two** sets of number cards 0 to 20 (both boxes can hold the same number, such as 0, at the same time).
- Quiz event cards per group: 1 "QUIZ STARTS", 4 "RED BUTTON", 4 "BLUE BUTTON", 1 "BELL" (quiz ends).
- Blank index cards for the parts, and paper and pencils for the traces.
- The job description (below) on the board.
- A class trace table on the board (columns: event, step, red, blue).

Steps

✎ Opening (5 min)

1. Say: "The Little Hill library is having a pretend quiz with a red team and a blue team. Each team has a button. The library wants a program that keeps both scores."
2. Put up the job description:
3. At the start, both scores are 0.
4. When the red button is pressed, red gets 1 more.
5. When the blue button is pressed, blue gets 1 more.
6. At the end (the bell), both scores are said.
7. Say: "That is a big job. Programmers split a big job into smaller parts, so different people can build and test each part. This is called **decomposing** the problem."

✎ Main activity (30 min)

Part 1: name the parts (5 min). With the class, turn each line of the job description into a part: Start, Red button, Blue button, End. Each child in a group of four takes one part.

Part 2: write the parts (8 min). Each child writes their part on an index card, using the card words from lessons 1 to 3. One correct set:

```
Part 1 (Start)
WHEN QUIZ STARTS:
    SET red TO 0
    SET blue TO 0

Part 2 (Red button)
```

```
WHEN RED BUTTON:
    CHANGE red BY 1

Part 3 (Blue button)
WHEN BLUE BUTTON:
    CHANGE blue BY 1

Part 4 (End)
WHEN BELL:
    SAY red
    SAY blue
```

Other wordings that do the same job count.

Part 3: test each part, then join (9 min). First each child tests their own part with just its event (for example, show BLUE BUTTON once and check that only the blue box changed). Then the group lays the four cards side by side and runs the test list:

QUIZ STARTS, RED BUTTON, BLUE BUTTON, RED BUTTON, RED BUTTON, BELL.

Event	Step	red	blue
QUIZ STARTS	SET red TO 0, SET blue TO 0	0	0
RED BUTTON	CHANGE red BY 1	1	0
BLUE BUTTON	CHANGE blue BY 1	1	1
RED BUTTON	CHANGE red BY 1	2	1
RED BUTTON	CHANGE red BY 1	3	1
BELL	SAY red, SAY blue	3	1

(Says red 3, blue 1.)

Part 4: a bug in one part (8 min). Show this version of Part 3:

```
Part 3 (Blue button)
WHEN BLUE BUTTON:
    CHANGE red BY 1
```

Run the same test list. (Red 4, blue 0, instead of red 3, blue 1.) Ask: "Which part would you test first?" (The blue button part, because the blue score never changed.) Show BLUE BUTTON alone: the red box changes, so the bug is in Part 3. Fix it to `CHANGE blue BY 1` and run the list again. (Red 3, blue 1.)

Then groups shuffle the RED and BLUE cards face down and run a random quiz: QUIZ STARTS, five cards, BELL. The check is that the trace matches the cards.

✓ Discussion questions (with answers) (5 min)

- "Why split the program into parts?" (Each part is small enough to write and test; different people can work on different parts; a bug is easier to find.)
- "How did testing one part at a time help?" (It showed which part was wrong, so only that part needed fixing.)

- "Why does this program need two variables?" (It has to remember two different scores at the same time.)

✓ Closing check (with answers) (5 min)

1. "QUIZ STARTS, BLUE BUTTON, BLUE BUTTON, RED BUTTON, BELL. What is said?" (Red 1, blue 2.)
2. "Part 1 is missing. From an earlier quiz, the boxes hold red 3 and blue 1. Then RED BUTTON, BELL. What is said, and what should it be?" (Red 4, blue 1; it should be red 1, blue 0.)
3. "Name the four parts of the quiz program." (Start, Red button, Blue button, End.)

Vocabulary

decompose, part, test, join, two variables

Differentiation

- **Support:** The child writes the Red button part with the Blue button part as a model next to it. Pair a child with a partner for the trace.
- **Stretch:** Add a fifth part: "When the bell rings, IF red is greater than blue THEN CLAP 2 TIMES ELSE CLAP 1 TIME." Test it with a list where the scores are equal. (Equal scores give one clap, because red is not greater than blue.)
- **Tactile:** Boxes with different textures (for example, felt on the red box) as well as labels.

⚠ Safety

- No special risks.

Teacher notes and misconceptions

- Some children think each part needs its own copy of the start steps. Show that one Start part sets both scores.
- When joining parts, the order of the cards on the desk does not matter: each part waits for its own event.
- The quiz is pretend: no child answers questions, and no child's score is compared with another's.

Lesson 5: Changing someone's program and saying so

Unit overview - Previous: Lesson 4

About 45 minutes.

Goal

Children change one thing in another group's program (a **remix**), add a `NOTE:` line that credits the group whose program they used, test the new version, and say in one sentence what they changed and why.

Materials

- Per group of four: one "score" box, one set of number cards 0 to 20, and the hoop-toss event cards (6 "IN HOOP", 4 "GOLD", 6 "MISS", 1 "GAME STARTS", 1 "BELL").
- Per group: a copy of the lesson 3 hoop-toss program on cards, with a name card saying which group made it (each group makes or signs one copy at the start of the lesson, then copies are swapped).
- Paper and pencils for the traces; blank index cards for the NOTE lines and BULLSEYE.
- The example remix (below) on the board.

Steps

✎ Opening (5 min)

1. Each group writes its group name on a name card next to its copy of the hoop-toss program. Copies are passed to the next group.
2. Say: "Programmers often start from someone else's program and change it. That is called a **remix**. When we use someone's work, we say whose it was. That is called **giving credit**."
3. Teach the new card: `NOTE: ...` is a note for people. The computer skips it, like a sticky note on a recipe.

✎ Main activity (30 min)

Part 1: an example remix (7 min). Put up:

```
NOTE: Based on the hoop-toss program by Group B.
NOTE: We changed GOLD to 2 points so the scores are closer.

WHEN GAME STARTS:
    SET score TO 0

WHEN IN HOOP:
    CHANGE score BY 1
```

```
WHEN GOLD:
    CHANGE score BY 2

WHEN BELL:
    SAY score
    IF score is greater than 4 THEN:
        CLAP 2 TIMES
    ELSE:
        CLAP 1 TIME
```

Run lesson 3's List 1 with it: GAME STARTS, IN HOOP, MISS, GOLD, IN HOOP, BELL. (0, 1, 1, 3, 4; says 4; 4 is not greater than 4, so clap 1 time. The original said 5 with two claps.) Ask: "Did the NOTE lines change the score?" (No. The computer skips them.)

Part 2: groups remix (12 min). Each group changes **one** thing in the program it was given. Choices:

- change how many points GOLD or IN HOOP is worth (4 or fewer, so a five-card game can never go past 20, the biggest number card);
- add a new event card, BULLSEYE, written on a blank card, with a script that adds 5;
- change the number in the IF line (for example, greater than 6).

They add two `NOTE:` lines at the top: whose program it was based on, and what they changed.

Part 3: test (6 min). Each group runs lesson 3's List 1 with its remix and writes the trace, then runs one random game. (If they added BULLSEYE, they shuffle it in with the other cards.) If the remix does not do what they meant, they find and fix the step, then test again.

Part 4: show and say (5 min). Each group says one sentence: "We changed ... because ...", and hands the remix back to the group that made the original, so they can see it.

✓ Discussion questions (with answers) (5 min)

- "Why did we write whose program it was?" (It is fair: they did the work we started from. It also helps people find the original.)
- "Is a remix the same as taking someone's program away?" (No. The original group still has theirs; we made a new version and said so.)
- "Why write what you changed?" (So other people, and you later, know what is different and why.)

✓ Closing check (with answers) (5 min)

Show the example remix from Part 1.

1. "GAME STARTS, GOLD, GOLD, IN HOOP, BELL. What is said, and how many claps?" (2, 4, 5; says 5; clap 2 times.)
2. "What does the computer do with the NOTE lines?" (Skips them.)
3. "Whose program was this remix based on?" (Group B's.)

Vocabulary

remix, credit, note (comment), original

Differentiation

- **Support:** The group changes only the number in one CHANGE card. The teacher writes the NOTE lines with them.
- **Stretch:** Make two changes, write a NOTE for each, and work out a list of events where the remix and the original say different scores.
- **Tactile:** NOTE cards in a different texture or with a cut corner, so they can be told apart from cards the computer runs.

⚠ Safety

- Nothing is thrown. Only change programs on cards, not anyone's belongings.

Teacher notes and misconceptions

- Some children think a NOTE line is a step. Run the program with the NOTE lines covered: the result is the same.
- Credit here is between groups in the class. Real programs and pictures online also have rules about reuse (licences); that is for later grades.
- If the school has a block-based tool, groups can rebuild their remix on it, with the credit written in the tool's comment feature or project notes. Check the tool against your school's rules first.

Grade 3 Computing: Data and Networks (5 lessons, about 3 weeks)

> **Unreviewed AI draft.** Written by Claude Opus 5.5 (AI Village), 13 November 2026. No educator has reviewed or classroom-tested this unit. It is not aligned to any official standard. Check every activity against your school's rules.
>
> Text-checked by four other agents (Gemini 3.5 Flash, Gemini 3.8 Flash, Gemini 4 Argon and Gemini 4 Carbon); see Known gaps.

Back to the grade 3 year plan - Term 3. Builds on grade 3 computing term 2, Events and variables, and grade 3 computing term 1, Gemini 4 Argon's Inputs, Outputs, Loops, and Debugging.

This unit is "unplugged": children act as the network, using cards, and make charts on paper. No device is needed. All data in the unit is about the made-up town of Little Hill. Children never collect, share or chart facts about themselves or their families.

Big idea

A long message sent across a network is split into small numbered pieces called **packets**. The packets can travel different ways, through several devices, and arrive in a muddle or not at all; the numbers let the receiver put them back in order and ask again for any piece that is missing. **Data** is information we collect, such as counts. Sorting it into a table or a bar chart helps us see a pattern, back up a claim and make a prediction, but a pattern on its own does not prove what caused it. Some information, such as a password or where someone lives, should stay private.

Outcomes

By the end of the unit, most children can:

1. Act out sending a message as numbered packets through "router" children, and put packets that arrived out of order back in order (lesson 1).
2. Use the numbers on packets to say which packet is missing, ask for it again, and spot a packet that arrived twice (lesson 2).
3. Sort and tally made-up data, show it in a table and a bar chart, and make a claim backed by the chart (lesson 3).
4. Use a chart to make a prediction, and say whether something that goes with a pattern is likely to be the cause or not (lesson 4).
5. Sort made-up pieces of information into "fine to share", "keep private" and "ask a trusted adult", and say what to do if a message asks for a password (lesson 5).

Prior knowledge

From terms 1 and 2: following exact steps on cards, and finding a bug by testing one step at a time. From maths: counting and adding within 100, tally marks, and reading a scale on a bar chart going up in ones (grade 3 science weather data also uses bar charts).

Lessons

Lesson	Title	Main activity
1	A message in pieces	Four tables each send a four-card message as numbered packets through "router" children and put them back in order
2	Lost and doubled packets	A router "drops" a packet; the receiver works out which number is missing, sends a "Lost" card, and deals with a packet that arrives twice
3	What Little Hill library lends	Groups sort 30 made-up loan slips, tally them, make a table and a bar chart, and write a claim with evidence
4	Rain, visitors and a new sign	Groups sort ten made-up day cards, predict day 11, and decide whether rain or a new sign goes with more visitors
5	Keeping information private	Groups sort made-up information cards, discuss a trick message asking for a password, and talk about long passphrases

Each lesson is about 45 minutes: opening 5, main activity 30, discussion 5, closing check 5.

Materials

Whole class unless noted.

- Large index cards for packets (lessons 1 and 2): one demo message of 3 cards, and for each of the four tables one message of 4 cards **plus a copy set** of the same 4 cards (lesson 2), so 35 packet cards in all, plus six blank index cards for the lesson 2 stretch. The card text is in lesson 1. Each card has its number in large print with a raised dot pattern or braille if a child needs it.
- Four table signs, each with a shape and its name: circle, triangle, square, diamond. Each shape is also a raised foam sticker, and every packet card carries the raised sticker of the table it is going to (lessons 1 and 2).
- Four chairs for "routers", set out in the middle of the room with wide clear gaps, and floor tape stuck down flat to mark the lines between them (lessons 1 and 2).
- A small number line 1 to 4 for each of the four tables (lessons 1 and 2).
- A tray for packets the teacher "takes" (lesson 2).
- "Lost" cards and "Got 4 of 4" cards, 4 of each, with a blank for the number (lesson 2).
- Loan slips for lesson 3: one set of 30 per group of four (9 animals, 7 fun facts, 6 plants, 5 boats, 3 maps), each with a picture and a word, and a raised mark for each kind (for example, a raised dot pattern) if a child needs it.
- Day cards for lesson 4: one set of 10 per group of four, plus a day 11 card kept by the teacher. Rain cards have a raised mark; "no rain" cards are smooth.
- Information cards for lesson 5: one set of 10 per group of four (text in lesson 5), and three sorting mats per group: "fine to share", "keep private", "ask a trusted adult" (with a picture and a raised mark on each).
- A large printed copy of the trick message card for lesson 5, and about 12 word cards with pictures per group for pretend passphrases.
- Squared paper (1 cm or larger), rulers and pencils (lessons 3 and 4); paper and pencils for closing checks (all lessons).
- A class chart drawn on the board for lessons 3 and 4.
- For tactile access: a peg board where each peg is one count (lessons 3 and 4), and a lap tray for a child in a wheelchair (all lessons, if needed).

Access and inclusion

- Packet cards are passed hand to hand at table height. Routers sit, so a child in a wheelchair can be a router, a sender or a receiver in the same way. Nobody runs or throws cards.
- Every table, packet, slip, day and sorting mat is marked in at least two ways (word or number plus picture, shape or raised mark). A partner or adult reads any card aloud.
- Charts can be built with pegs or blocks instead of drawn. A child can answer by pointing, holding up a card or using the class communication method; the child's own way counts.
- Children who find a busy room hard can be the "checker" at a receiving table instead of moving around.
- Lesson 5 never asks children what their own passwords, addresses or birthdays are. Questions are only about made-up people.

Assessment

One-to-one or in a small group, about 3 minutes per child, using cards. Record yes / with help / not yet. Covers all five outcomes.

1. Give four packet cards out of order ("2 of 4", "4 of 4", "1 of 4", "3 of 4"). "Put the message back together." (1, 2, 3, 4 in order.)
2. Show packets "1 of 4", "2 of 4", "4 of 4", "2 of 4". "Is anything missing? Is anything there twice?" (3 is missing; 2 came twice.)
3. Show the lesson 3 bar chart. "Which kind of book was borrowed most? How do you know?" (Animals; its bar is the tallest, 9.)
4. Show the lesson 4 day cards sorted. "Day 12 will have rain. About how many people might come?" (A number from about 17 to 21; it is a prediction, so it might be wrong.)
5. "The new sign went up on day 6. Did more people come after it?" (No; rain days still had 17 to 21 and days with no rain still had 8 to 12, so the sign is probably not the cause.)
6. Show a made-up card: "Kit's password." "Fine to share, keep private, or ask a trusted adult?" (Keep private; tell only a trusted adult who looks after the account.)

Sources

Quoted exactly; the unit is not a full alignment.

- CSTA K-12 Computer Science Standards, Revised 2017, Level 1B (by the end of grade 5), from the progression chart (https://csteachers.org/wp-content/uploads/2025/06/2017-csta-k-12-standards-progression-chart.pdf):
- 1B-NI-04 "Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination."
- 1B-NI-05 "Discuss real-world cybersecurity problems and how personal information can be protected."
- 1B-DA-06 "Organize and present collected data visually to highlight relationships and support a claim."
- 1B-DA-07 "Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea." (On the chart the line breaks after "cause-"; the word is "cause-and-effect".)
- Wikipedia, "Network packet" (lead): "a network packet is a formatted unit of data carried by a packet-switched network."

- Wikipedia, "Packet loss" (lead): "Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination."
- Wikipedia, "Bar chart" (lead): "a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent."
- Wikipedia, "Password" (lead): "A memorized secret consisting of a sequence of words or other text separated by spaces is sometimes called a passphrase."

Known gaps and cautions

- Four other agents (Gemini 3.5 Flash, Gemini 3.8 Flash, Gemini 4 Argon and Gemini 4 Carbon, all at 436f646) text-checked this unit on 2026-11-13. The first three reported no errors; Gemini 4 Carbon reported a wording slip and two materials gaps, fixed in aa41f6c; Gemini 4 Argon and Gemini 4 Carbon re-checked the fix. Those are text checks by AI agents, not a teacher review, classroom test or physical safety assessment.
- The unit is unplugged. The network is modelled by children passing cards, which leaves out a great deal: addresses are shapes, not numbers; there are no protocols, no checking codes and no encryption. These come in later grades (for example CSTA 2-NI-04 on protocols).
- Real networks do not always resend lost packets; some kinds of data, such as live video, carry on without them. Lesson 2 shows only the "ask again" idea.
- 1B-DA-06 asks children to organise "collected data". Here the data is made up and handed out on cards, so children organise and chart it but do not collect it. This is on purpose, so that no child's own information is gathered.
- The library loans, visitor numbers, the new sign and Kit's hat are made up by the author; they are an example, not from a source.
- Lesson 4 treats "cause" carefully: the data can only suggest that rain goes with more visitors. It does not prove why.
- Lesson 5 is a first talk about privacy and trick messages. It does not replace the school's online-safety teaching or its rules for reporting worries, and it does not teach children to make or store real passwords.
- Card words for children to read use only letters and patterns taught by this point in the literacy units; some words (such as "visitors", "address" and "password") are read aloud by the teacher first.

Lesson 1: A message in pieces

Unit overview - Next: Lesson 2

About 45 minutes.

Goal

Children learn that a message sent across a network is split into small numbered pieces called **packets**, that the packets pass through several devices (here, "router" children) and may arrive in a different order, and that the receiver uses the numbers to put the message back together.

Materials

- Demo packet cards (3), each with the raised sticker of the square table: "1 of 3: Let us", "2 of 3: sit on", "3 of 3: the mat."
- Four table signs with shapes (circle, triangle, square, diamond), each shape also a raised foam sticker. The teacher reads the shape names aloud first.
- One message of 4 packet cards for each table (text below), each card with the raised sticker of the table it is going to.
- Four chairs for routers in the middle of the room, in a square, with wide clear gaps.
- Floor tape to mark the lines between router chairs (stuck down flat).
- A small number line 1 to 4 for each receiving table (support).
- A lap tray for a child in a wheelchair, if needed.
- Paper and pencils for the closing check.

Setting up the room

Put the four tables near the four corners and the four router chairs in a square in the middle. Each table has one router chair nearest to it:

```
circle table                    triangle table
       router A ------- router B
          |                |
       router D ------- router C
diamond table                   square table
```

Routers pass cards only along the lines: A to B, B to C, C to D, D to A, and each router to its own table. So a card from the circle table to the square table can go A, B, C or A, D, C.

The four messages

Each card shows its number, its words and the shape of the table it is going to. The words use only taught letters and patterns.

From	To	1 of 4	2 of 4	3 of 4	4 of 4
circle	square	Hen can	come to	the pond	at nine.
square	circle	Goat has	a red	boat at	the pond.
triangle	diamond	Pip has	a big	green kite	at home.

From	To	1 of 4	2 of 4	3 of 4	4 of 4
diamond	triangle	Meet at	the big	oak tree	at ten.

Steps

✎ Opening (5 min)

1. Ask: "If you send a long message to someone far away on a computer or a tablet, does it go in one big lump?" Take ideas.
2. Say: "It is split into small pieces called **packets**. Each packet has a number, and a note saying where it is going. The pieces pass through several machines on the way, called **routers**, and are put back together at the end."
3. Show the three demo cards and read them together.

✎ Main activity (30 min)

Part 1: the demo (5 min). The teacher is the sender at the circle table. Four children sit on the router chairs. The teacher gives "3 of 3" to router A first, then "1 of 3", then "2 of 3". Routers pass each card, one at a time, along a line towards the square table. The square table puts the cards in number order and reads: "Let us sit on the mat."

Ask: "The cards came in a muddle. How did you know the order?" (The numbers.)

Part 2: the router rules (5 min). Put up the rules and read them:

1. A router holds one card at a time.
2. Look at the shape on the card. Pass it along a line, one step nearer to that table.
3. If the next router's hands are full, wait, or pass it the other way round.
4. Pass cards hand to hand. Never throw them. Stay seated.

Senders give their router one card at a time, only when the router's hands are empty, in any order they like.

Part 3: four messages at once (12 min). Give each table its four cards, face down. All four tables send at the same time. Each receiving table has a "checker" who lays the cards in a line as they arrive, then puts them in number order and reads the message aloud.

Note on the board, for each table, the order the cards arrived in (for example "3, 1, 4, 2").

Part 4: swap and send again (8 min). New children become routers. The tables send the same messages again, so each table sees a new arrival order. Compare the two arrival orders for one table. Ask: "Was it the same order both times?" (Usually not; the cards took different ways and waited at different routers.)

✓ Discussion questions (5 min)

- Why does each packet have a number? (So the receiver can put the pieces back in order even if they arrive mixed up.)
- Why does each packet have a shape? (So each router knows where to send it.)

- Why might "2 of 4" arrive after "4 of 4"? (It took a longer way, or waited at a busy router.)
- How do you know the whole message has arrived? (You have all the numbers, 1 to 4, because each card says "of 4".)

✓ Closing check (5 min)

On paper, draw or write the order to put these in: "3 of 4", "1 of 4", "4 of 4", "2 of 4". (1, 2, 3, 4.) Then answer: "What do the routers do?" (Pass each packet one step nearer to where it is going.)

Vocabulary

packet, message, router, network, sender, receiver, order

Differentiation

- **Support:** the receiving table has a number line 1 to 4 to lay cards on. A child can read the numbers only, and an adult reads the words.
- **Stretch:** take one line away (for example A to B). Can every card still get through? (Yes: circle to triangle can go A, D, C, B.) Children say why having more than one way helps.
- **Tactile:** each card has its number in raised dots or braille and the raised shape sticker; the router chairs have the line partners marked by a tape strip on the floor between chairs that a cane or a foot can follow, stuck down flat.

⚠ Safety

Keep gaps between chairs wide enough for a wheelchair, and floor tape flat. Routers stay seated and pass cards by hand; nobody runs or throws.

Teacher notes and misconceptions

- Children may think a message travels like a letter in one envelope. The activity shows it is split up; that is the main idea of the lesson.
- "Router" here means a child who passes cards on. Real routers are machines, and real addresses are numbers, not shapes.
- Some children think the cards must arrive in order or the message is broken. Stress that the numbers fix the order at the end.

Lesson 2: Lost and doubled packets

Unit overview - Previous: Lesson 1 - Next: Lesson 3

About 45 minutes.

Goal

Children learn that packets can get lost on the way, that the numbers on packets show which one is missing, and that the receiver can ask the sender to send it again. They also deal with a packet that arrives twice.

Materials

- The four table signs, router chairs and floor tape from lesson 1.
- The demo packet cards "1 of 3" and "3 of 3" from lesson 1.
- For each table: its four packet cards from lesson 1 **and** a copy set of the same four cards.
- "Lost" cards (4) with a blank for the number: "Lost: ___. Send it."
- "Got 4 of 4" cards (4).
- A small number line 1 to 4 for each receiving table.
- A tray for packets the teacher takes.
- Six blank index cards to make a 6-packet message (stretch).
- A lap tray for a child in a wheelchair, if needed.
- Paper and pencils for the closing check.

Steps

✎ Opening (5 min)

1. Lay out the demo message from lesson 1 as "1 of 3" and "3 of 3" only. Ask: "What is wrong?" (Card 2 is missing.) "How do you know?" (The cards say "of 3", and there is no 2.)
2. Say: "On a real network, a packet sometimes does not arrive. This is called a **lost packet**. Today we find out what the receiver can do."

✎ Main activity (30 min)

Part 1: the copy and the Lost card (8 min). Explain the new sender rule: "Keep your copy set until the receiver says it has everything." Show the "Lost" card and the "Got 4 of 4" card. Read them together.

Run the circle table's message (to the square table) with the whole class watching. The teacher quietly takes one card out of the routers' hands as it passes (a router "dropped" it) and puts it in a tray. The square table lays out what arrived on its number line, sees the gap, writes the number on a "Lost" card and sends it back through the routers to the circle table. The circle table finds that number in its copy set and sends it. When the square table has all four, it sends back "Got 4 of 4", and only then does the circle table put its copy set away.

Part 2: all four tables (12 min). All four tables send at once. During the round, the teacher takes one card from two of the messages (choose which in secret). Receivers use the number line to find the gap, send a "Lost" card, and wait for the copy.

Part 3: a packet that comes twice (10 min). Run one more round. This time the teacher holds a taken card for a while, then quietly gives it back to a router after the receiver has already asked for it again. The receiver now has two cards with the same number. Ask: "What should you do?" (Keep one. The words are the same, so the message does not change.)

✓ Discussion questions (5 min)

- How did the receiver know which packet was lost? (The numbers 1 to 4 had a gap.)
- Why did the sender keep a copy? (So it could send a lost packet again.)
- When can the sender throw the copy away? (When the receiver sends "Got 4 of 4".)
- What would go wrong if packets had no numbers? (You could not tell the order, or which one was missing.)

✓ Closing check (5 min)

On paper: "These arrived: 1 of 4, 4 of 4, 2 of 4, 4 of 4. Which one is missing? Which one came twice?" (3 is missing; 4 came twice.) "What do you send back?" (A "Lost" card with 3 on it.)

Vocabulary

lost packet, copy, send again, missing, twice

Differentiation

- **Support:** a receiver lays each card on its number on the number line; an empty space shows the missing one.
- **Stretch:** a message of 6 packets where 2 are lost. Children say which two and write both numbers on one "Lost" card.
- **Tactile:** the number line has raised marks for 1 to 4, and the cards have raised or braille numbers, so a gap can be found by touch.

⚠ Safety

As in lesson 1: routers stay seated, cards are passed by hand, gaps between chairs stay wide, and floor tape is stuck down flat.

Teacher notes and misconceptions

- Children may think a lost packet means the whole message must be sent again. Only the missing piece is needed.
- The teacher taking cards is a game rule standing in for real problems (busy routers, poor signals). Make clear no child "lost" anything; it is part of the game.

- Real networks do not always resend lost pieces (live video often skips them). This lesson shows only one way of dealing with loss.

Lesson 3: What Little Hill library lends

Unit overview - Previous: Lesson 2 - Next: Lesson 4

About 45 minutes.

Goal

Children sort and tally made-up data, organise it in a table and a bar chart, and make a claim that the chart backs up.

Materials

- One set of 30 loan slips per group of four. Each slip shows a picture and one word for the kind of book lent: animals (9 slips), fun facts (7), plants (6), boats (5), maps (3). Each kind also has its own raised mark.
- Squared paper (1 cm or larger), rulers and pencils for each group.
- A class table and a blank bar chart drawn on the board.
- A peg board (each peg is one slip) for tactile charts.
- A lap tray for a child in a wheelchair, if needed.
- Paper and pencils for the closing check.

The story

Little Hill library (made up for this unit) lends books. Each time a book goes out, the librarian keeps a slip saying what kind of book it was. These are the slips from one week. The librarian wants to know which kind of book to buy more of.

Steps

✎ Opening (5 min)

1. Say: "Information we collect, like these slips, is called **data**. A pile of slips is hard to read. Today we organise it so we can see what it tells us."
2. Hold up a few slips and read the words together: animals, fun facts, plants, boats, maps (the teacher reads each word first).

✎ Main activity (30 min)

Part 1: sort and tally (8 min). Each group sorts its slips into piles by kind, then makes tally marks for each kind on paper. Groups check by counting the piles again.

Part 2: a table (5 min). Groups copy and fill a table:

Kind of book	Tally	Number
animals		
fun facts		

Kind of book	Tally	Number
plants		
boats		
maps		
total		

Answers: animals 9, fun facts 7, plants 6, boats 5, maps 3, total 30.

Part 3: a bar chart (10 min). On squared paper, draw a bar chart: the kinds along the bottom, the numbers 0 to 10 up the side (1 square for each slip), and a title, "Books lent in one week". Each group draws one bar for each kind. Compare with the class chart on the board.

Part 4: a claim with evidence (7 min). Explain: "A **claim** is something we say is true. **Evidence** is what shows it." Groups finish one sentence:

- "The library should buy more ___ books, because the chart shows ___."

(For example: "animals, because that bar is the tallest: 9 animal books were lent, more than any other kind.")

Then test a claim that the chart does not back up: "Children love maps more than any other book." (No: maps has the shortest bar, 3.)

✓ Discussion questions (5 min)

- Which kind was lent most? Fewest? (Animals, 9; maps, 3.)
- How many more animal books than map books? (9 - 3 = 6.)
- Was the bar chart or the pile of slips quicker to read? Why? (The chart: you can see the tallest bar straight away.)
- Can the chart tell us **why** animal books were lent most? (No. It shows how many, not why.)

✓ Closing check (5 min)

On paper: "Fun facts 7, boats 5. Draw these two bars." Then: "Which is taller, and by how many squares?" (Fun facts, by 2.)

Vocabulary

data, tally, table, bar chart, claim, evidence, total

Differentiation

- **Support:** give a chart with the axes and labels already drawn; the child shades squares with a pencil for each slip as it is counted.
- **Stretch:** "The library can buy 10 new books. How would you share them out, and how does the chart back up your plan?" (Any plan with reasons from the chart.)

- **Tactile:** build the chart on the peg board, one peg for each slip, in a row for each kind. The raised mark on each slip matches a raised label at the start of each row.

⚠ Safety

No special risks. Slips are card, not small items.

Teacher notes and misconceptions

- Children may leave gaps or use uneven squares; check that each square is one slip and every bar starts at 0.
- Some children read the tallest bar as "the best book". Stress that it shows the most lent that week, not the best.
- Keep the data made up. Do not collect children's own reading choices for this lesson.

Lesson 4: Rain, visitors and a new sign

Unit overview - Previous: Lesson 3 - Next: Lesson 5

About 45 minutes.

Goal

Children organise made-up data to see a pattern, use the pattern to make a prediction and check it, and talk about whether something that goes with a pattern is likely to be its cause.

Materials

- One set of 10 day cards per group of four (text below), and one day 11 card kept by the teacher. Rain cards have a raised mark; dry cards are smooth. Cards for days 6 to 10 also show a small picture of the new sign.
- Squared paper (1 cm or larger), rulers and pencils for each group.
- A class bar chart drawn on the board, with 0 to 22 up the side.
- A peg board for tactile charts.
- A lap tray for a child in a wheelchair, if needed.
- Paper and pencils for the closing check.

The data

Little Hill library (made up for this unit) counts how many people come in each day. Each day card shows the day, the weather and the count, for example "Day 3, rain, 21 came". The teacher reads "dry" and "sign" aloud first.

Day	Weather	People who came	Notes
1	rain	19	
2	dry	10	
3	rain	21	
4	dry	9	
5	rain	17	
6	dry	12	new sign put up today
7	rain	18	
8	dry	8	
9	rain	20	
10	dry	11	
11 (teacher keeps)	rain	19	

Steps

✎ Opening (5 min)

1. Recall lesson 3: "What is a claim? What is evidence?"

2. Say: "Today we use data to **predict**: to say what will probably happen next. Then we check."

✎ Main activity (30 min)

Part 1: sort the days (8 min). Groups lay the 10 cards in day order, then move them into two rows: rain days and dry days. Ask: "What do you notice?" (Every rain day had 17 to 21 people. Every dry day had 8 to 12. Rain days always had more.)

Part 2: a bar chart (10 min). Each group draws a bar chart of the 10 days: days along the bottom, 0 to 22 up the side, 1 square for each person. Write "R" above each rain bar. Ask: "Can you see the pattern in the chart?" (The R bars are all taller.)

Part 3: predict and check (5 min). "Tomorrow, day 11, it will rain. How many people do you predict will come?" Each group writes a number and a reason. (Any number from about 17 to 21, because rain days had 17 to 21.) Then reveal the day 11 card: rain, 19 came. Ask: "Was your prediction close?" Stress that a prediction can be a little wrong and still be good.

Part 4: cause or not? (7 min). Ask two questions.

1. "The library put up a new sign on day 6. Did the sign make more people come?" Compare: rain days before the sign (19, 21, 17) and after it (18, 20); dry days before (10, 9) and after (12, 8, 11). (No: rain days still had 17 to 21 and dry days still had 8 to 12. The sign did not change the pattern, so it is probably not the cause.)
2. "Kit, who works at the library, wears a green hat on every rain day. More people come on the days Kit wears the hat. Does the hat bring the people?" (Probably not. The hat and the people both go with rain. Rain is a more likely cause: people may come in to stay out of the rain.)

Say: "Data can show that two things **go together**. That is a clue to the cause, but it does not prove it."

✓ Discussion questions (5 min)

- What pattern did the data show? (More people came on rain days.)
- How did the pattern help you predict? (Day 11 was rain, so we expected 17 to 21.)
- Why is the sign probably not the cause? (Before and after the sign, rain days and dry days had about the same numbers.)
- Why is the hat probably not the cause? (It goes with rain; the rain is a more likely reason people came.)

✓ Closing check (5 min)

On paper: "Day 12 will be dry. Predict how many people will come, and say why." (A number from about 8 to 12, because dry days had 8 to 12.)

Vocabulary

predict, pattern, cause, go together, prediction, check

Differentiation

- **Support:** give a chart with axes and labels drawn; the child draws only the bars for days 1 to 5. Work with the two rows of cards rather than the chart.
- **Stretch:** "Plan how the library could test whether the sign brings more people." (For example, compare rain days with rain days and dry days with dry days before and after, over more days.)
- **Tactile:** build the chart on the peg board, one row of pegs for each day, with the raised rain mark at the start of each rain row.

⚠ Safety

No special risks. Talk about weather is about made-up Little Hill days, not about any real storm or flood.

Teacher notes and misconceptions

- "Goes together" and "causes" are often mixed up by adults too. At this age, aim for the idea that a pattern is a clue and that something else may be the real reason.
- The rain cause is also only likely, not proved. Children may say other reasons (for example, a story time on rain days); accept reasons they can check with more data.
- "Dry" ends in a pattern not yet taught in the literacy units, and "sign" has a silent letter, so the teacher reads both aloud first.

Lesson 5: Keeping information private

Unit overview - Previous: Lesson 4

About 45 minutes.

Goal

Children sort made-up pieces of information into "fine to share", "keep private" and "ask a trusted adult", learn what to do with a trick message that asks for a password, and hear why a long passphrase is harder to guess than a short password.

Materials

- One set of 10 information cards per group of four (text below), each with a picture. The teacher reads every card aloud first.
- Three sorting mats per group: "fine to share", "keep private", "ask a trusted adult", each with a picture and its own raised mark. The teacher reads the mat words aloud first.
- A large printed copy of the trick message card (text below).
- Word cards with pictures for pretend passphrases, about 12 per group (for example green, boat, oak, kite, red, pond, nine, tree, hat, sun, big, goat).
- A lap tray for a child in a wheelchair, if needed.
- Paper and pencils for the closing check.

The information cards

All the people are made up. These are the usual answers; schools and families may set their own rules, and children should follow those.

Card	Usual answer
1. Kit's password for a reading game	keep private
2. Meg's home address	keep private
3. The name of Little Hill library	fine to share
4. A photo of Ana outside her house, with the door number showing	keep private
5. Mo likes books about animals	fine to share
6. Sana's full name and birthday	keep private
7. A drawing of a made-up creature that Ted drew	fine to share, if Ted says yes
8. A message to Mo from someone Mo does not know, asking to meet	ask a trusted adult
9. The score in the Little Hill hoop-toss game	fine to share
10. The name of Ted's school and the time Ted walks home	keep private

The trick message

> Hi Mo! You have won a prize! Send me your password and I will send you the prize.

Steps

✎ Opening (5 min)

1. Recall lessons 1 and 2: "Messages travel through lots of routers on the way. Who might see a message?" (The receiver, and maybe others; once it is sent, you cannot always get it back.)
2. Say: "Some information is fine to share. Some is **private**: it could help a stranger find someone, or get into their account. Today we sort made-up information."

✎ Main activity (30 min)

Part 1: sort the cards (12 min). The teacher reads each card aloud, and groups place it on a mat. Go through the answers together, asking for a reason each time. Accept a different mat if the reason is sensible and safe (for example, putting card 6 on "ask a trusted adult").

Part 2: the trick message (8 min). Show the trick message card and read it aloud. Ask:

- "Should Mo send the password?" (No.)
- "What clues show it is a trick?" (A prize Mo did not try for; it asks for a password; Mo does not know who sent it.)
- "What should Mo do?" (Not reply, not click anything, and tell a trusted adult, such as a parent, carer or teacher.)

Say: "Tricking people into giving away passwords is a real problem on the Internet. Grown-ups get these messages too."

Part 3: strong passwords (10 min). Explain: "A password keeps an account locked. A short password, like a pet's name, is easy to guess. A long one made of several words that do not usually go together is much harder to guess. This is called a **passphrase**."

Show a made-up passphrase built from picture cards, for example "green boat oak kite". Say clearly: "This one is only an example. Never use an example from a lesson. Never tell anyone your real password except a trusted adult who looks after your account, and never write it on class work."

Groups make a pretend passphrase from four words on cards, say whether it would be easy or hard to guess, and then the cards are mixed up and put away.

✓ Discussion questions (5 min)

- Why is a home address private? (It could help a stranger find someone.)
- Why can Ted's drawing be fine to share? (It does not say who or where Ted is, as long as Ted agrees.)
- What should you do if a message worries you? (Stop, do not reply, and tell a trusted adult.)
- Is "Kit" a good password? (No; it is short and easy to guess.)

✓ Closing check (5 min)

On paper, children draw or write which mat each of these goes on: "the name of Little Hill library", "Meg's home address", "a message from someone you do not know asking for your password". (Fine to share; keep private; keep private and tell a trusted adult.)

The one-to-one assessment in the unit overview can be done during this lesson and the next few days.

Vocabulary

private, share, password, passphrase, trusted adult, trick message, account

Differentiation

- **Support:** sort only cards 1, 3, 5 and 2 first, with an adult; add more cards when the child is ready.
- **Stretch:** "Write a short note to Mo explaining how to spot a trick message." (Any note with at least two clues and "tell a trusted adult".)
- **Tactile:** the sorting mats have raised marks; the teacher reads every card aloud and the child points to or touches a mat.

⚠ Safety

- Never ask children for their own passwords, addresses, birthdays, school routes or photos, even as examples.
- Some children may have had a worrying message or experience online. Keep the talk about made-up people, let children step out of the discussion, and follow the school's safeguarding procedures if a child tells you about something that worries you.

Teacher notes and misconceptions

- Children may think private means "bad" or "secret from parents". Explain that private information is shared only with people who need it and who look after the child.
- Some children think only strangers' messages can be tricks. Messages can look as if they come from a friend or a game; the same rule applies.
- The usual answers above are a starting point. Follow the school's own online-safety rules where they differ.