LESSON 6

The four steps of computational thinking, by building

40 minutes | Strand 1: Introducing computer science | Outcome 1.6

WHAT THIS LESSON DOES

Apply the four steps of computational thinking to plan and create your own Scratch project. Choose a build idea such as an animated scene or quiz, decompose it, recognise patterns, abstract details and write an algorithm before coding it.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.6	construct code to implement algorithms

WHAT STUDENTS SHOULD BE ABLE TO DO

- Use decomposition, pattern recognition, abstraction and algorithms to break down a build you choose, naming each step as you use it
- Turn the plan into a working Scratch program (LO 1.6)

BEFORE THE BELL

- Write the four steps on the board and leave them visible throughout the lesson.
- Prepare a simple chase example to demonstrate each step aloud as you reach that bullet (name, plain meaning, chase line), not as a full list up front.
- Decide the exact save method you will announce in the last two minutes (cloud save, File → Save to computer into the class folder, or your usual route).
- Student devices with Scratch and earlier projects ready to open
- Student devices with Scratch; decide today's exact save method before the period

The four steps of computational thinking, by building

HOW THE LESSON RUNS

4 min	introduction	Start
6 min	content	Four steps, one idea
2 min	activity	Choose your build idea
5 min	content	Write your four-step plan
5 min	content	Build the first piece and run it
8 min	activity	Keep building from your plan
4 min	content	Debug what breaks, then save
3 min	content	Make sense
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Keep the four step names displayed and refer to them often.
- At five minutes into the first activity, call plans down and open Scratch even if algorithms are rough.
- Circulate during activities and ask students which step they are using.
- Use one quick predict question before a first green-flag run: what do you expect to see?
- Use student examples in the Make sense section to reinforce vocabulary.

WHAT TO WATCH FOR

- Students treat the steps as a strict linear sequence instead of revisiting them. Remind them the steps can be used iteratively.
- Plans consist only of drawings without naming the steps. Require students to label each part with the step name; offer four pre-written headings if needed.
- Students focus on decoration rather than functional code. Redirect to completing the algorithm and one clear working behaviour first.
- Students equate abstraction with "first version only". Correct gently: abstraction means keep only what matters and set aside irrelevant detail.

DIFFERENTIATION

- Support: Give the student four headings already written (Decomposition / Pattern recognition / Abstraction / Algorithms) to fill in with one short line each.
- Support: Pair students who need help with a partner for the planning phase.
- Extension: Challenge students who finish the minimum bar early to add the next piece of the plan and refine the algorithm, naming which step justified the change.