LESSON 6

The Four Steps of Computational Thinking, by Building

40 minutes | Think Like a Programmer

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.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Apply the four steps of computational thinking to break down and plan a build.
- Implement the plan as a working Scratch program.

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).

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	content	Choose Your Build Idea
5 min	content	Write Your Four-step Plan
5 min	content	Build the First Piece and Run It
8 min	content	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.