technology

Coding sequences and debugging in Scratch

Lesson at a glance

Hook pupils with the finished Banana Jump scene and the idea that programmers test and debug. Model each Scratch piece on the IWB — Blue Sky backdrop, running cat, banana clones moving along the ground, space-bar jump, and collision stop — while pairs copy, run the green flag after every piece, and fix bugs as they appear. Close with play-testing and a hands-off code talk on one bug they found and one change they would make next time.

Learning objectives

- Build a Scratch program by putting the blocks in the right order so each sprite does what you want
- Find and fix a bug in a program and describe what was wrong

Before the bell — prep

Book devices and open scratch.mit.edu on the IWB before the bell. Have your own Banana Jump project ready to model piece by piece. Check pairs can share a keyboard; note any login or network snags early.

Materials

Item	Qty	Per	Source	Low-cost substitute
tablet or laptop with Scratch	1	pair	school devices	run the lesson on the IWB with pupils taking turns

Safety watch-point

Shared devices and keyboards — fair turns, calm handling, and save before closing. No physical hazard beyond normal classroom screen use.

Lesson 26

Teaching moves

- **What we're building:** Show the finished scene on the IWB. Ask what we call it when a program fails and we fix it — land on debugging. Set the routine: watch the model, copy, run the green flag, share the keyboard calmly.
- **Create a new project:** Model a brand-new project so everyone starts clean. Scan for pairs still on an old file and have them start fresh before any blocks go down.
- **Add the Blue Sky backdrop:** Show the backdrop library and point to the brown strip as the ground the cat will stand on. Confirm every pair has Blue Sky, not a lookalike, before moving on.
- **Animating the Cat:** Model go-to x:-142 y:-110 plus the forever next-costume loop, then run so the class sees the cat low-left and running. If a cat floats or vanishes, check the coordinates together. Ask what makes the run look continuous.
- **Add the bananas sprite:** Add Bananas from the library on the IWB. From here on, pause and name which sprite is selected before any new blocks — wrong-sprite code is the main later bug.
- **Adding Bananas:** Model the hide-and-shrink setup and stress that the banana is meant to vanish on green flag. Reassure pairs who think nothing happened; visible movement comes next. Ask why a random wait between bananas helps the game.
- **Making Bananas Move:** Model the clone-movement stack and run so bananas travel right to left along the ground. If clones appear in the wrong place or stay hidden, check start position and the show block with the pair.
- **Jumping Cat:** Model the space-key jump on the cat sprite and test on the IWB. Confirm the stack is on the cat, not the bananas. Ask what brings the cat back down; invite a quick change to repeat count or change-y with a prediction first.
- **Detecting Collisions:** Model the touching check as a forever 'watchman' on the cat. Test that a hit stops the game. If it never stops, the check is on the wrong sprite or outside the loop — fix that together and have pupils say what they see when it works.
- **Wrap up:** Give pairs play-test time. Circulate, note one or two strong fixes with names, and prompt gentle tinkering — higher jump, more bananas — while watching what breaks. Remind everyone to save.
- **Code talk — what worked and what we'd change:** Hands off keyboards, faces front. Draw out one specific bug and how the pair tracked it, then restate their steps in fuller sentences for the class. Invite two or three 'one thing we'd change' ideas. Keep it short and conversational.

What it should show

A working Banana Jump: cat runs low-left, jumps on space, banana clones drift along the ground, and touching a banana stops the game. Common odd results — cat off-screen (wrong coordinates), bananas never appear (missing show/clone start), jump does nothing (block on wrong sprite or wrong key), game never ends (touching check not on the cat inside a forever loop). Re-check sprite selection and run after each fix.

Misconceptions & interventions

- **Nothing happened — the banana is broken because I can't see it after green flag.** — Remind them the banana is deliberately hidden and small at first; the visible moving clones come in the next stacks. Run your model to the clone-move step so they see the plan.
- **I put the jump blocks on and space does nothing.** — Check which sprite is selected — the jump must sit on the cat, triggered by space. Click the cat, confirm the hat block, and retest on the IWB.
- **The cat hits a banana but the game keeps going.** — Trace the touching check: it belongs on the cat inside a forever loop. Move or wrap the block with them and run until a hit stops the script.

Differentiation

- {'emerging': ['Sit support pairs at the teacher table and let them run and observe each modelled piece before copying; keep sprite-name checks verbal at every step.', 'Provide the coordinate and key names aloud so they match the IWB exactly rather than typing from memory.'], 'developing': ['After the game works, change wait time or jump height, predict the effect, then run and say what changed.', 'Describe one bug they fixed in a full sentence during code talk.'], 'proficient': ['Tinker with more bananas or a higher jump and explain what broke and how they restored a fair playable game.', 'Offer one concrete improvement for a rebuild and justify it from how the current scripts behave.']}

Cross-curricular hook

Links to Maths sequencing and position: ordered steps, x/y coordinates, and predicting how a number change alters movement.