Bug Hunt Debug – Fixing Mistakes in Outdoor Algorithms

Curriculum Links

Scotland

I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way
TCH-13

UK

use logical reasoning to predict the behaviour of simple programs

Key Concept

I can follow a sequence of instructions, spot errors, and correct them using logical thinking.

4 QUALITY EDUCATION

13 CLIMATE ACTION

Background

Debugging is a key part of computational thinking. It involves identifying errors in a sequence and fixing them so a process works as intended. In this energetic, nature-based lesson, learners become real-life debuggers, following simple “code” trails in an outdoor space—then analysing what went wrong, correcting it, and reflecting on how errors affect outcomes.

Resources

- Chalk, cones, hoops, or sticks to create visible paths or instruction points
- Printed or verbal "programs" for learners to follow (sequences of movements)
- Error cards (optional – simple cards with things like “turn left” when it should be right)

Wider Skills

- Problem-solving – Spotting and correcting mistakes
- Observation – Noticing where instructions don’t match reality
- Teamwork – Working together to identify and fix bugs
- Communication – Explaining logic clearly

Metaskills

- Sense-making – Understanding how instructions connect to real-world movement
- Judgement – Deciding where and why a bug happened
- Self-management – Staying focused through trial and error
- Decomposition – Breaking down the algorithm into parts to isolate the bug

Lesson

Introduction (10 minutes) – What is Debugging?

- Gather learners and ask:
  - What happens if a computer program has a mistake?
  - What if a robot is told to walk into a tree?
- Introduce the idea of a bug (a mistake in a set of instructions), and explain that debugging is how we find and fix it.
- Explain that today, they'll be following instructions to move around outside—but some instructions will have bugs!

Main Activity (30–40 minutes) – Debugging in Action
Part 1: Buggy Sequences (15–20 minutes)

- In pairs or small groups, learners take turns being the "robot" and the "programmer."
- One follows a pre-set path created by the teacher (or another team), using commands like:
  - Step forward x3
  - Turn right
  - Pick up the cone
  - Turn left
  - Step forward x2
- BUT… there is one or two bugs in the instructions (e.g., wrong direction, step missing).
- As learners follow the buggy algorithm, they must:
  - Notice where it goes wrong
  - Explain what the bug is
  - Rewrite or retell the corrected version

Part 2: Create Your Own Buggy Code (15–20 minutes)

- Learners create a short movement sequence using cones or natural markers.
- They write or say the "code" to another team – including one deliberate mistake.
- The other team follows the code and then debugs it, explaining what went wrong and how to fix it.

Plenary (10 minutes) – Reflecting on the Process

- Ask the class:
  - What was tricky about debugging?
  - How did you spot mistakes?
  - What made some bugs harder to find?
  - Why do you think debugging is important – not just in computing, but in life?

Key Discussion

- Key discussion questions:
- What made a bug obvious or hard to spot?
- How do good instructions help people or computers work better?
- Can we use debugging in everyday life? (e.g., fixing routines, instructions, games)

Assessment

Bug identification and explanation:
Learners should accurately spot where a sequence failed and explain why using logical reasoning.

Problem-solving and correction:
Learners should suggest or demonstrate an improved version, showing they understand how to fix the issue.

Team collaboration:
Observe whether learners support each other, take turns, and listen respectfully during the process.

Reflection and understanding:
In discussion or written work, learners show awareness of how debugging improves outcomes and how errors are opportunities to learn.

Follow-up task (optional):

- Learners write their own real-world instructions (e.g., how to make a sandwich, how to get to the garden)—with one bug—for a friend to debug.
- Extension: Introduce loops or if-then commands into their algorithms for an added challenge.

Notes

- Keep bugs fun and manageable—just one or two per sequence.
- Use this lesson as a precursor to basic coding lessons in the classroom.
- Encourage learners to think about real-world “bugs”—like giving unclear directions to a friend or writing an unclear set of instructions.
- Works brilliantly in mixed-ability pairs!

Younger Classes

- Use very simple movement sequences (e.g., Step forward, turn, pick up a stick).
- Make debugging more visual—e.g., use toy animals or cones to show what the robot “crashed into.”
- Focus on fixing one-step bugs, and celebrate each fix!

Older Classes

- Introduce conditional logic:
- If there is a rock, turn right. Otherwise, keep going.
- Let them map out their program in symbols or arrows and compare it to the result.
- Encourage peer-to-peer feedback: How could that instruction have been clearer?