Lesson Plan: Bug Hunters

Year Group: 7 | **Duration:** 50 minutes | **Topic:** Debugging

---

1. Overview

Core Concept: Debugging — systematically tracing through an algorithm to find and fix errors.

Learning Objectives:

- Identify three types of algorithmic bugs: wrong order, missing step, logic error
- Trace through a buggy algorithm to find where it fails
- Explain the fix and why it works
- Understand that systematic tracing is more reliable than intuition

Key Vocabulary:

Term	Definition
Bug	An error in an algorithm or program
Debug	Find and fix a bug
Wrong order	All steps are present but in the wrong sequence
Missing step	An essential step has been left out
Logic error	A condition or calculation is incorrect (e.g., < instead of >)
Trace	Follow an algorithm step by step, tracking all values
Test case	A specific input used to check if an algorithm works correctly

---

2. Before the Lesson

Print:

- ☐ `worksheet-buggy-algorithms.md` — 1 copy per student

No other materials needed.

---

3. Timed Lesson Flow

0–5 min — Hook: The Buggy Sandwich

1. Act out making a sandwich — but with bugs:
   - Put the filling between the bread slices first
   - Then spread butter on the outside of the bread
   - Then eat it... before cutting it

2. Students spot what went wrong.
3. Ask: *"What type of error was each? Were any steps missing? Were any in the wrong order?"*

5–10 min — Three Bug Types

Write on the board with examples:

1. **Wrong Order** — all steps are there, but in the wrong sequence
   - Example: "Eat toast → Wait for toaster → Put bread in toaster"
2. **Missing Step** — an essential step is absent
   - Example: "Add water → Stir → Drink coffee" (missing: add coffee!)
3. **Logic Error** — a condition or value is wrong
   - Example: `IF temperature < 100: boil the water` (should be `> 100` ... wait, that's wrong too. Should be: water boils AT 100, so this logic is wrong)

10–35 min — Individual Work

Students work through all 3 buggy algorithms on the worksheet:

- For each: trace through, identify where it fails, name the bug type, write the fix

35–42 min — Pair Comparison

Compare findings with a partner. Discuss any differences in diagnosis.

42–48 min — Class Share-Out

Teacher asks for different diagnoses. Discuss: can the same algorithm have multiple valid fixes?

48–50 min — Debrief

- Professional programmers spend around 50% of their time debugging
- Systematic tracing beats guessing every time
- The best way to find a bug: *follow every step exactly as a computer would*

---

4. Teacher Facilitation Notes

What to look for:

- Students who skim-read instead of tracing — insist on step-by-step execution
- Students who fix the "obvious" problem without checking if there are more bugs
- Students who fix the same bug in different ways (multiple valid fixes are fine!)

How to intervene minimally:

- *"Follow it exactly as a computer would. What is the output of step 3?"*
- *"You said the bug is at step 4. What would happen if you ran steps 1-3 perfectly — do they work correctly?"*
- *"Your fix changes step 4. Does the rest of the algorithm still work with that change?"*

Common misconceptions:

- Bugs are always obvious — many logic errors produce outputs that look almost correct
- One bug per program — algorithms can have multiple bugs
- Debugging means rewriting from scratch — debugging means finding the minimum fix

---

5. Extension / Early Finisher Tasks

1. **Write your own:** Create a 4th buggy algorithm for a partner to debug. Make the bug subtle.
2. **Multiple bugs:** Write an algorithm with TWO bugs of different types. Can your partner find both?
3. **Test cases:** For Algorithm 3, write 3 test cases that would catch the bug. Write 1 test case that would NOT catch it.

---

6. Key Takeaway

> **Debugging requires systematic tracing — follow every step exactly, check every value, never assume. A bug is not a mistake; it's a problem to solve methodically.**