COMPUTING · Y3-Y6

Algorithms

Knowledge Organiser — KS2 Computing

Key vocabulary

1. **Algorithm**
   A precise set of step-by-step instructions to solve a problem or complete a task.
2. **Sequence**
   The order that instructions happen in. Order matters in algorithms — getting it wrong changes the outcome.
3. **Decomposition**
   Breaking a big problem into smaller, easier parts.
4. **Debugging**
   Finding and fixing mistakes (bugs) in an algorithm or program.
5. **Bug**
   A mistake or error in code that makes it not work properly.
6. **Logical reasoning**
   Thinking carefully and step-by-step to predict what an algorithm will do.
7. **Pseudocode**
   Writing an algorithm in plain English-like language before turning it into actual code.
8. **Flowchart**
   A diagram that shows the steps of an algorithm using shapes and arrows.

is an algorithm?

Examples from everyday l

- A RECIPE is an algorithm — step-by-step instructions to make a meal.
- BRUSHING YOUR TEETH is an algorithm — wet brush, add toothpaste, brush, rinse, spit.
- GETTING DRESSED is an algorithm — and the order matters! Socks before shoes, not after.
- Algorithms are everywhere. Computers run thousands every second.
- An algorithm doesn't have to use a computer. Knitting patterns, board game rules, IKEA furniture instructions — all algorithms.

sequence matters

The order of steps changes eve

- Wrong sequence: 'Put on shoes. Put on socks.' (impossible!)
- Right sequence: 'Put on socks. Put on shoes.'
- Wrong sequence in cooking: 'Take cake out of oven. Put cake in oven.' (raw cake!)
- In computing, the wrong sequence means a program does the wrong thing — even if every individual step is correct.
- Always test your algorithm by walking through it step-by-step.

ecomposition

Break a big problem into small

- BIG PROBLEM: Make a class assembly.
- DECOMPOSED:
- 1. Choose a topic.
- 2. Write the script.
- 3. Make the props.

- 4. Practise.
- 5. Perform.
- Each smaller part is easier to work on.
- Computer programs use the same trick — break a complex task into small functions or modules.

Debugging

Finding and fixing bugs

- Step 1: TEST your program. Did it do what you wanted?
- Step 2: If not, find where it went wrong (the bug).
- Step 3: Read your code carefully — line by line.
- Step 4: Make a small change (one thing at a time).
- Step 5: Test again.
- Step 6: Repeat until it works.
- Most professional programmers spend MORE time debugging than writing new code.
- The first computer bug (1947) was a real moth stuck in a computer at Harvard — the term stuck.