Kit's Debugging Masterclass: Mastering the Fix

Learning objective: To understand the process of debugging in computing, identifying errors and refining algorithms to ensure a program functions as intended.

Use these flashcards to test your knowledge of debugging. Work through the cards with a partner, explaining your reasoning for the deeper questions.

What is a 'Bug' in computing?

A bug is an unexpected error or mistake in a computer program that prevents it from working correctly.

Define 'Debugging'.

Debugging is the systematic process of finding, analysing, and fixing errors within an algorithm or program.

What is a 'Logic Error'?

A logic error occurs when the code runs, but it does not produce the result the programmer intended because the instructions are in the wrong order or incorrect.

What is a 'Syntax Error'?

A syntax error is a mistake in the 'grammar' or structure of the code, such as a missing bracket or a spelling mistake, which stops the computer from understanding the command.

Why is it important to 'predict' what a program will do before running it?

Predicting allows a programmer to test their mental model of the algorithm against the actual outcome, making it easier to spot where a bug might be hiding.

If Kit the Fox writes an algorithm for a robot to move, but the robot walks into a wall, what is the first step of debugging?

The first step is to isolate the specific line of code or instruction where the sequence went wrong by testing parts of the program individually.

Compare and contrast a 'Syntax Error' and a 'Logic Error'.

A syntax error usually stops the program from running entirely (like a typo), whereas a logic error allows the program to run but results in an incorrect outcome (like a wrong instruction).

Explain why 'commenting' or annotating your code is helpful for debugging.

Adding notes explains the purpose of each section, making it easier to re-read your work later to find where your intended logic failed.

Justify why a programmer should change only one thing at a time when debugging.

Changing only one variable at a time helps you identify exactly which change fixed the problem; if you change too many things at once, you won't know which one was the successful solution.

What might happen if you do not debug your program?

The program will remain unreliable, inefficient, or completely non-functional, meaning it cannot reliably perform the task it was designed for.

If your code is perfect, can it still have a bug?

Yes, if the algorithm itself is flawed (e.g., trying to solve a maths problem with the wrong formula), the program will work perfectly but give the wrong answer.

Create a 'buggy' instruction for making a sandwich. How would you debug it?

Example: 1. Put bread on plate. 2. Eat sandwich. 3. Put jam on bread. Debug: The sequence is wrong. The jam must be applied before eating.