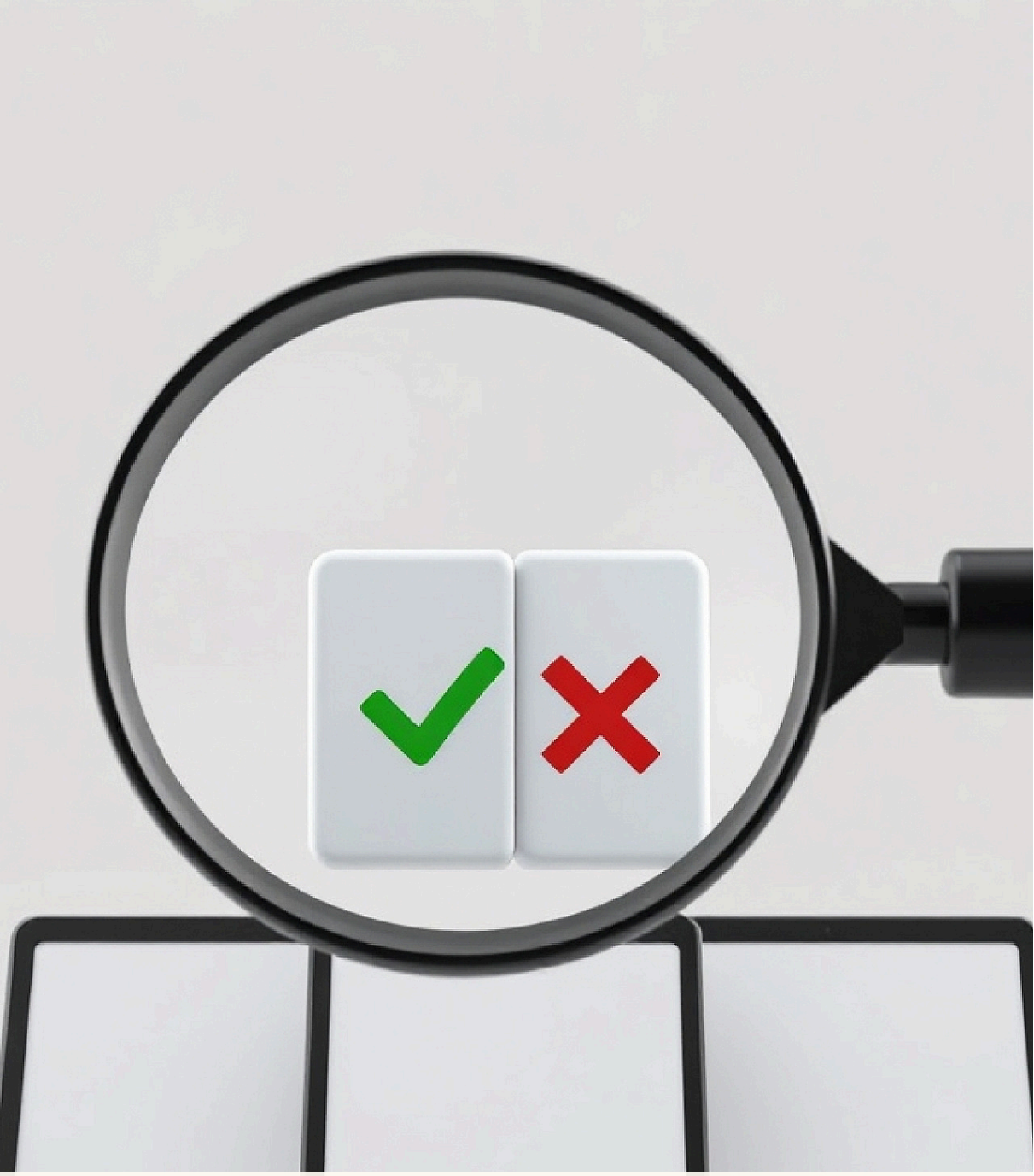
What is a condition?

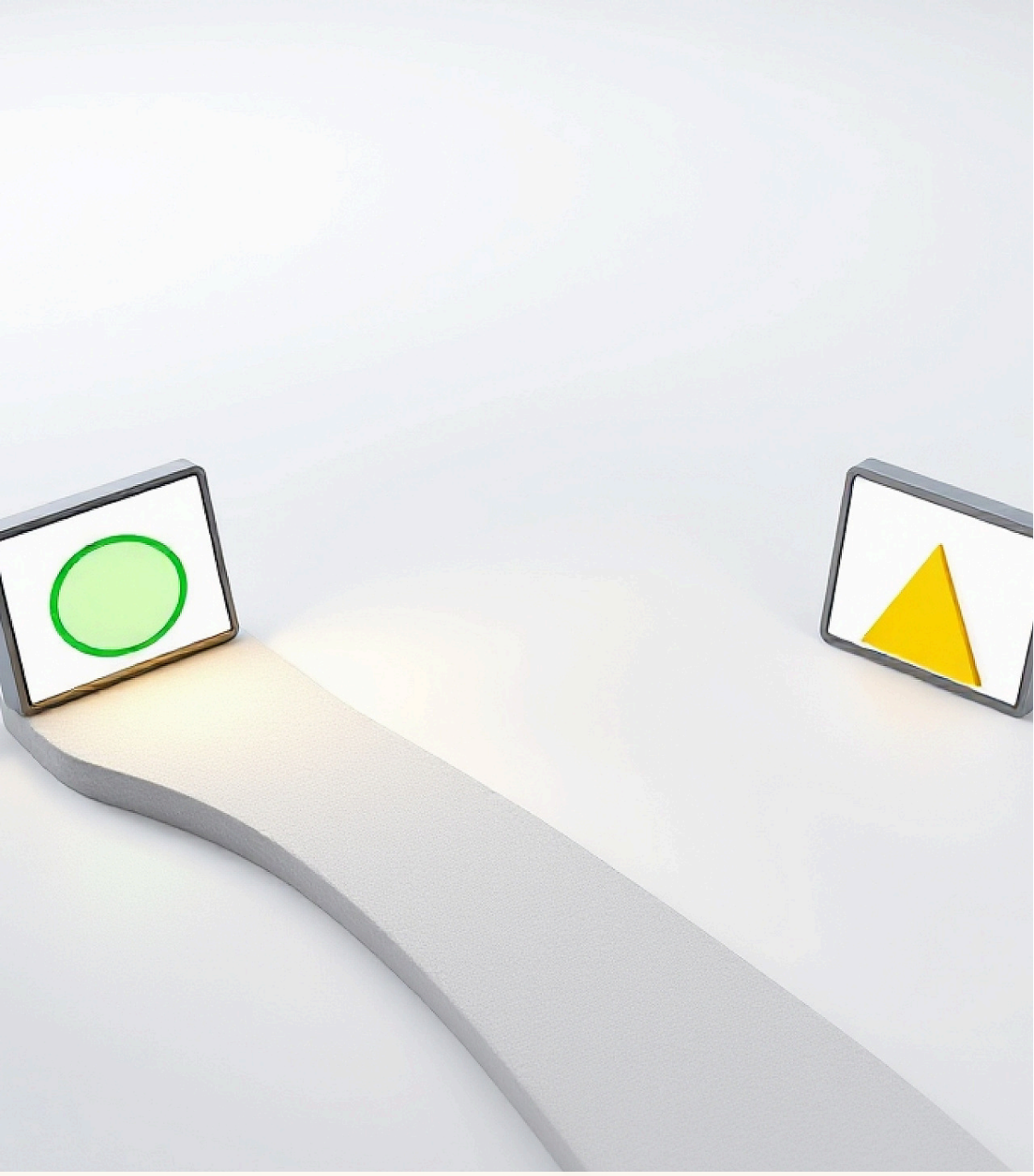


www.yzokul.com.tr

What is a condition?



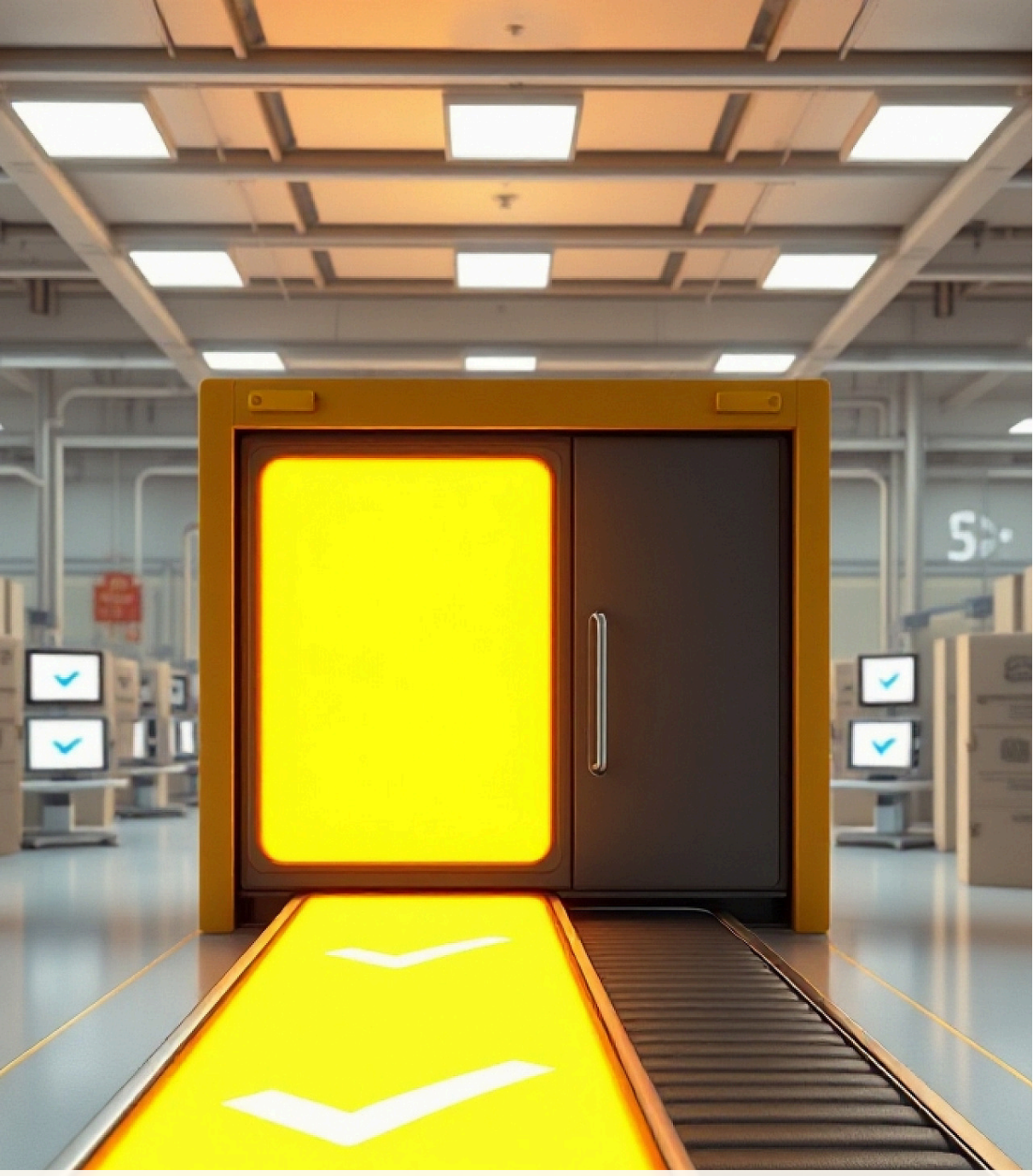
A condition is something a program checks to decide what to do next. You can think of it as a question the program asks about the current situation. The answer to that question is always either true or false. Based on the answer, the program chooses one action or another. Conditions are the basic tool that lets programs make decisions.



Programs do not always do the same thing every time they run. Sometimes a program needs to look at a situation and pick one of several possible actions. For example, a game might need to check whether the player has won or lost. A condition is how the program examines that situation before choosing what to do.



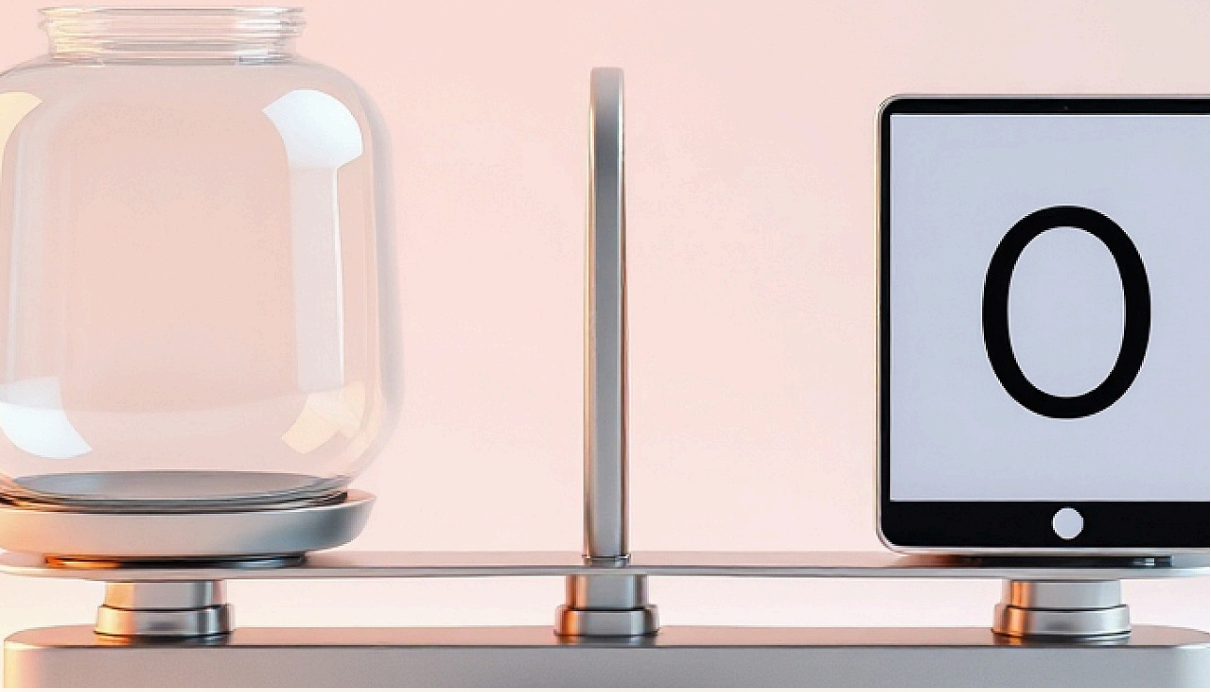
Picture a game where the player has health points. When health drops to zero, the game should end. If health is still above zero, the game should keep going. The program needs a way to check the health value and react correctly. That check is a condition, and it decides whether the game stops or continues.



Before choosing an action, the program checks the situation. That check is the condition, and its result decides the next step. The program does not guess or act randomly; it evaluates the condition first. Only after the result is known does it pick the matching action. The check always comes before the choice.



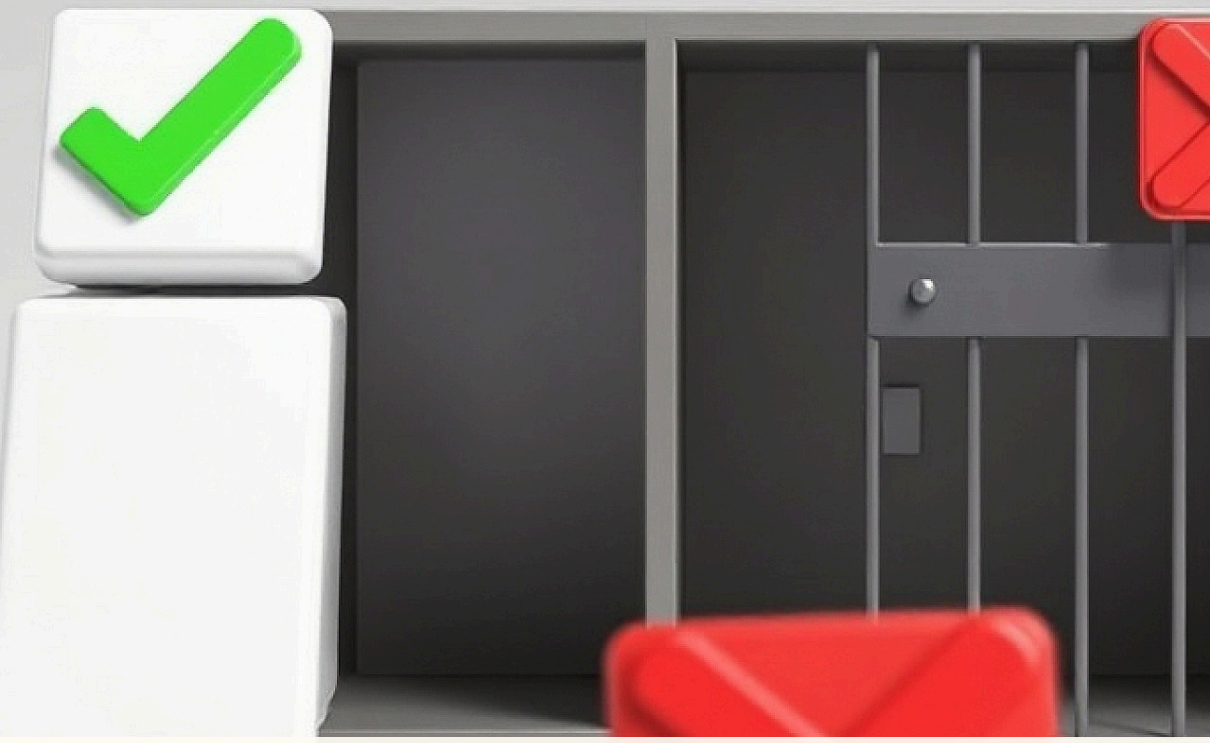
A condition is an expression that comes out as exactly one of two values: true or false. There is no in-between result and no maybe. The program treats true and false as the only possible answers. This simplicity is what makes conditions reliable for making decisions. Every condition you write will produce one of these two values.



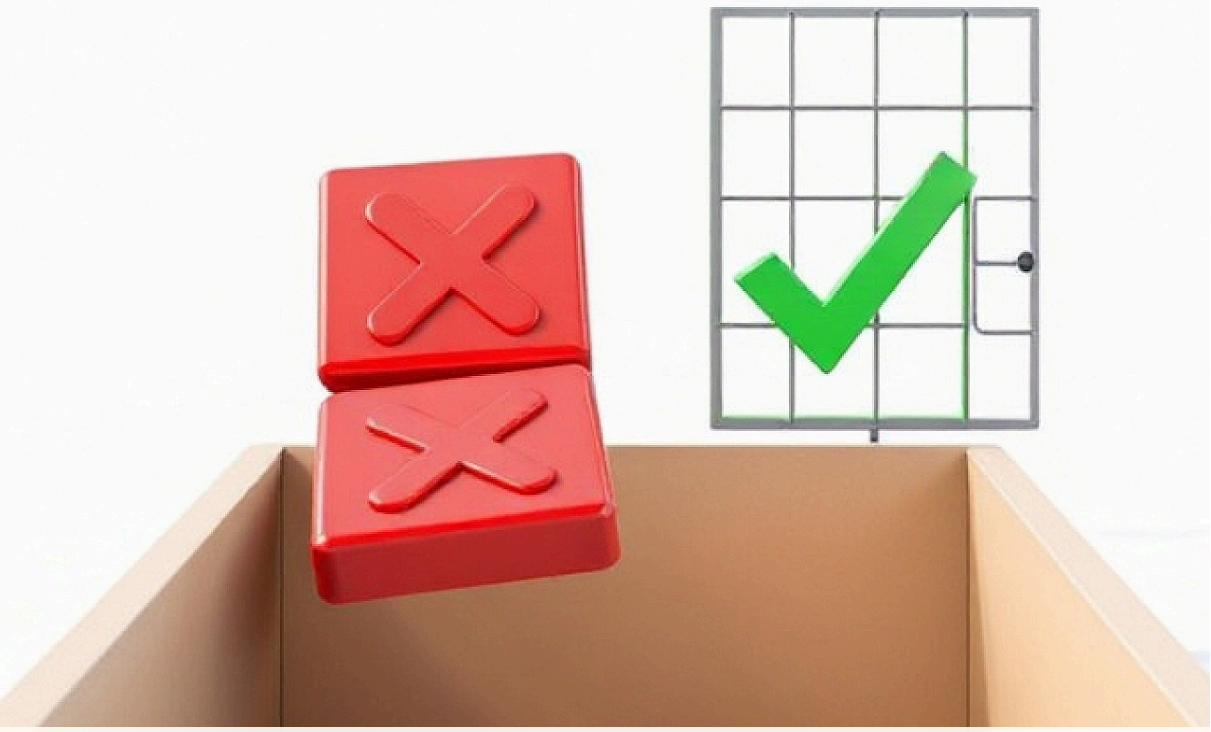
The expression health equals zero is a condition. If health is zero, the condition is true. If health is above zero, the condition is false. The program compares the current health value against zero to get this result. This comparison is a simple but powerful way to check a situation.



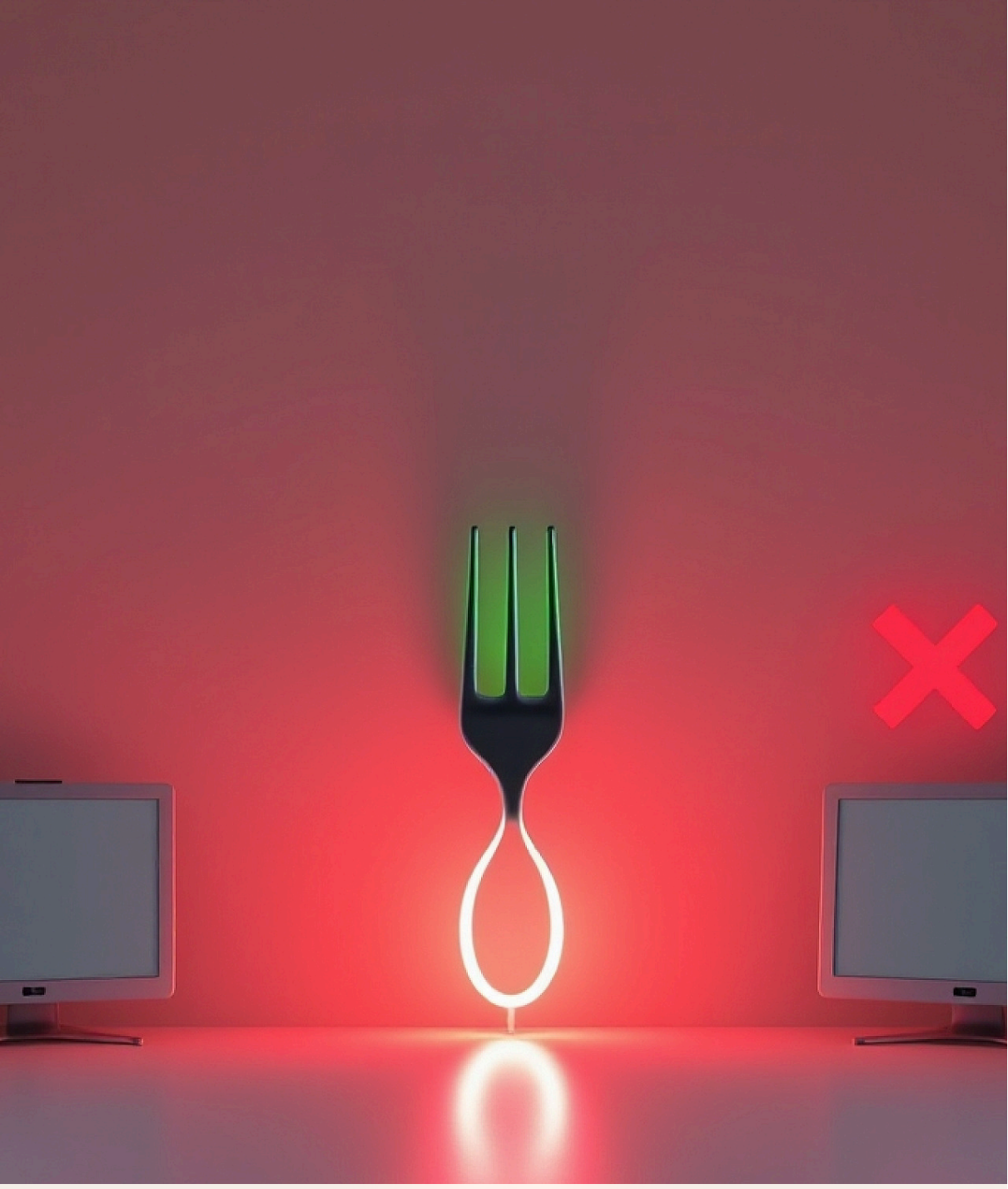
Programs use a structure meaning if to act on a condition. It says: if the condition is true, do this; if not, do something else. The if structure is the main tool for turning a condition into a decision. You write the condition once, and the program uses its result to choose a path. This keeps the decision clear and organized.



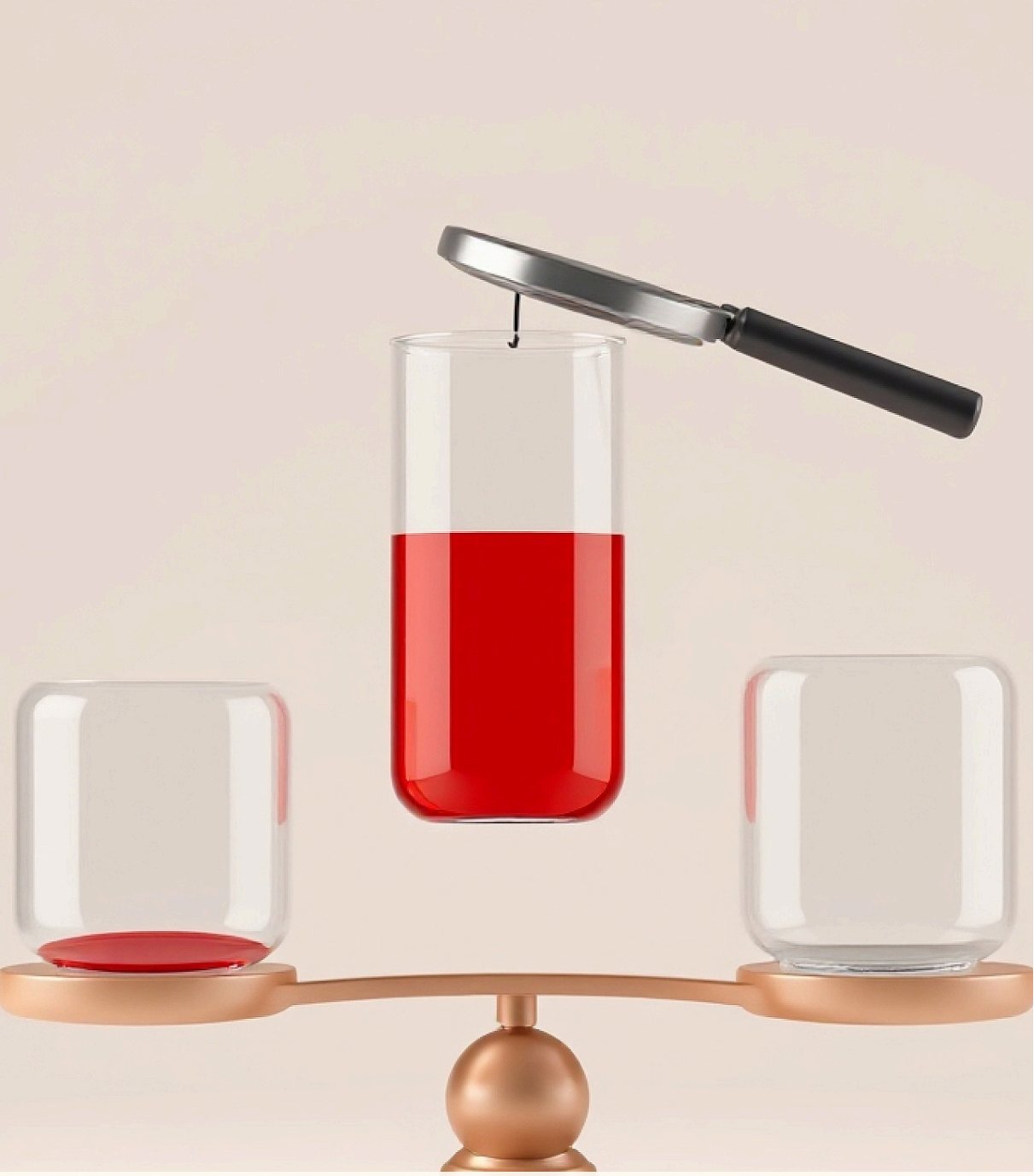
When the condition is true, the program runs the matching action. In the game example, true means health has reached zero, so the game ends. The program does exactly what the if structure says for the true case. It does not run the false action at the same time. Only one path is taken.



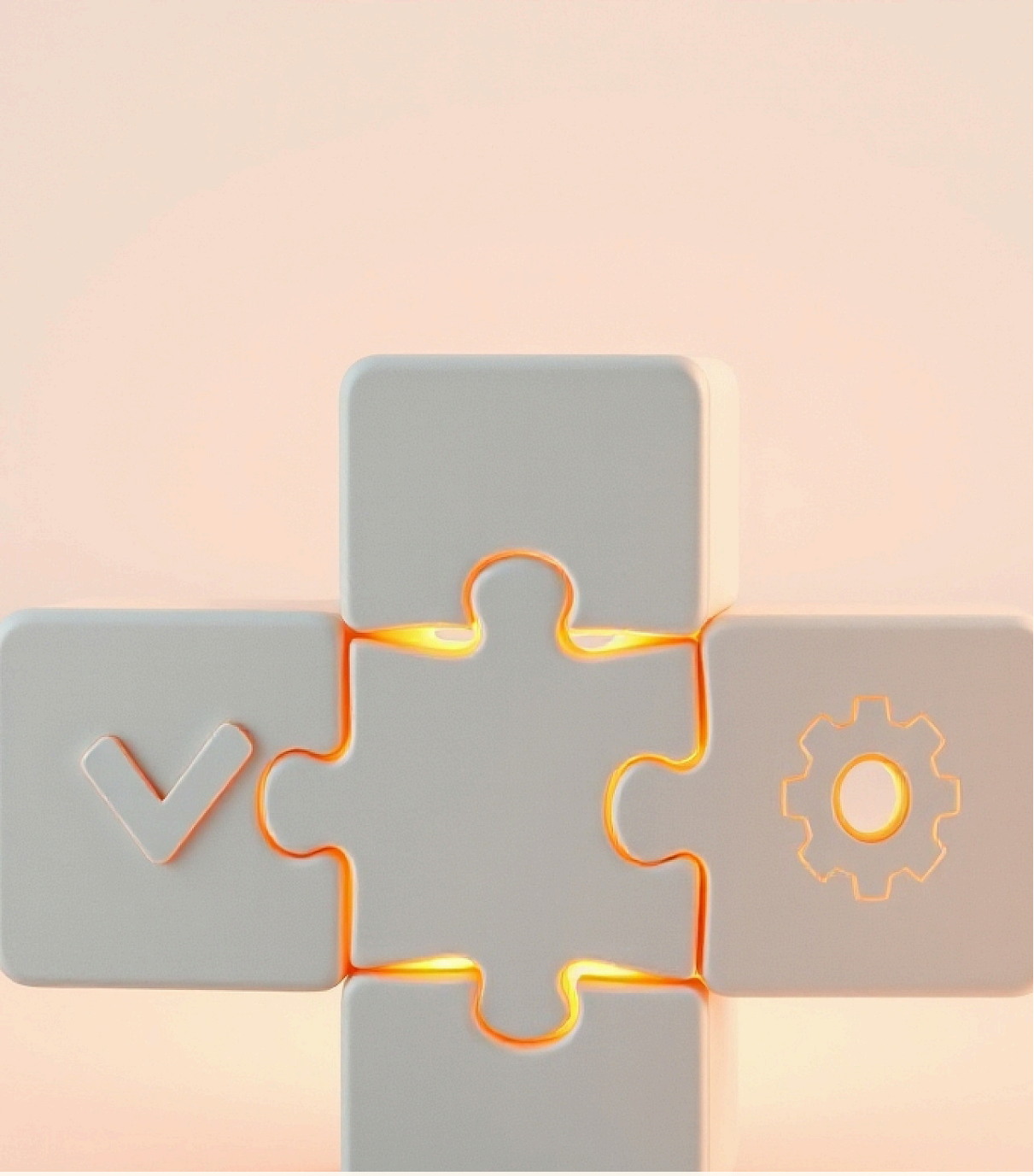
When the condition is false, the program skips the true action and does something else. In the game example, false means health is still above zero, so play continues as normal. The program ignores the ending action and follows the other path. This is how a single check can lead to different outcomes. The false path is just as important as the true path.



One condition splits the program into two possible paths. The program follows only the path that matches the true or false result. It never follows both paths at once. This is called a decision because the program chooses based on the condition. Two paths, one check: that is the core idea of using conditions.



Many conditions come from comparing two values, like health against zero. The comparison produces the true or false result that the if uses. You can compare numbers, text, or other values depending on what the program needs. The comparison itself is the condition. Keep the comparison simple and clear so the result is easy to predict.



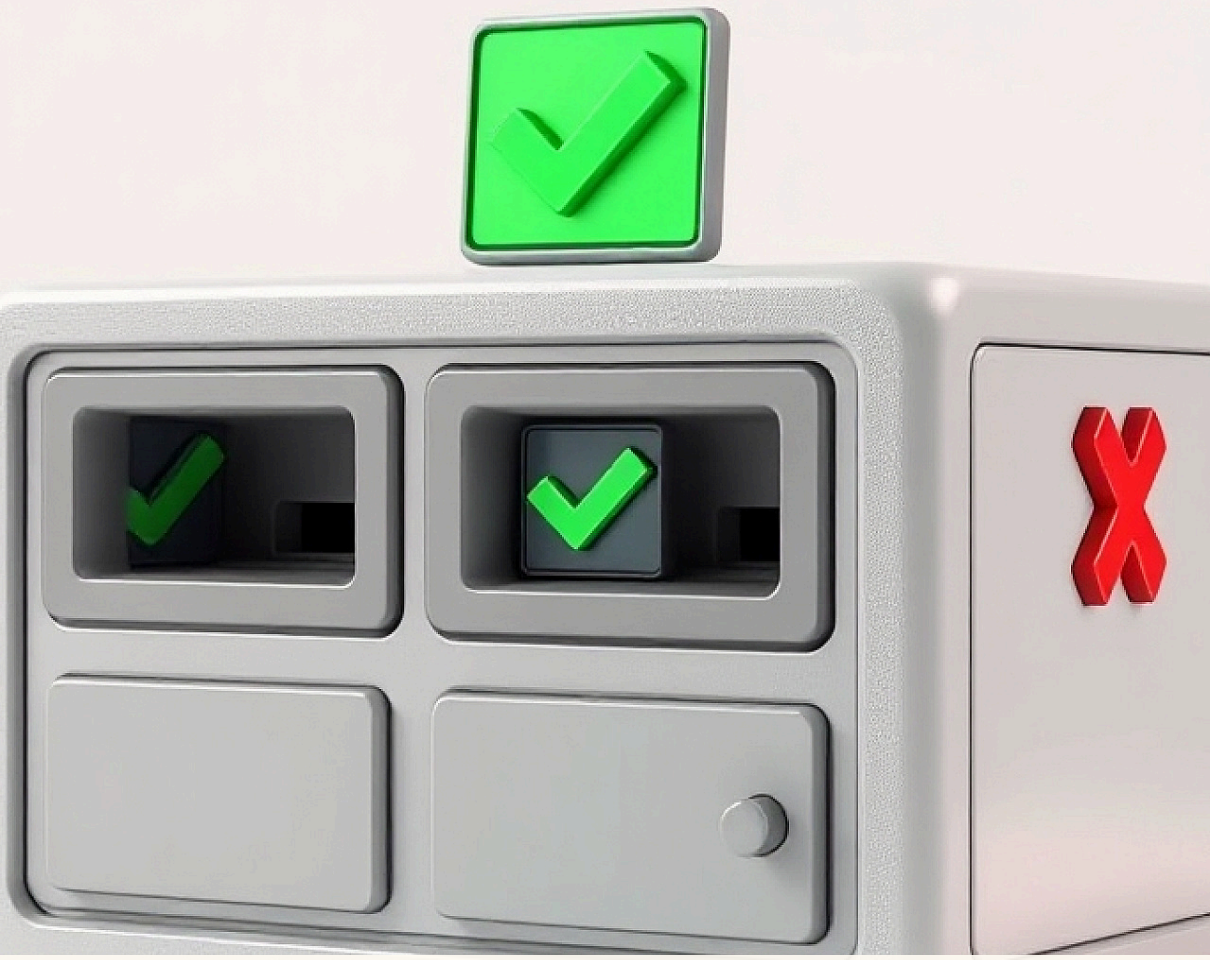
Several conditions can be joined together to make a more detailed decision. For example, end the game only if health is zero and no lives remain. Joining conditions lets you check more than one thing at once. The combined result is still a single true or false value. This gives you finer control over what the program does.



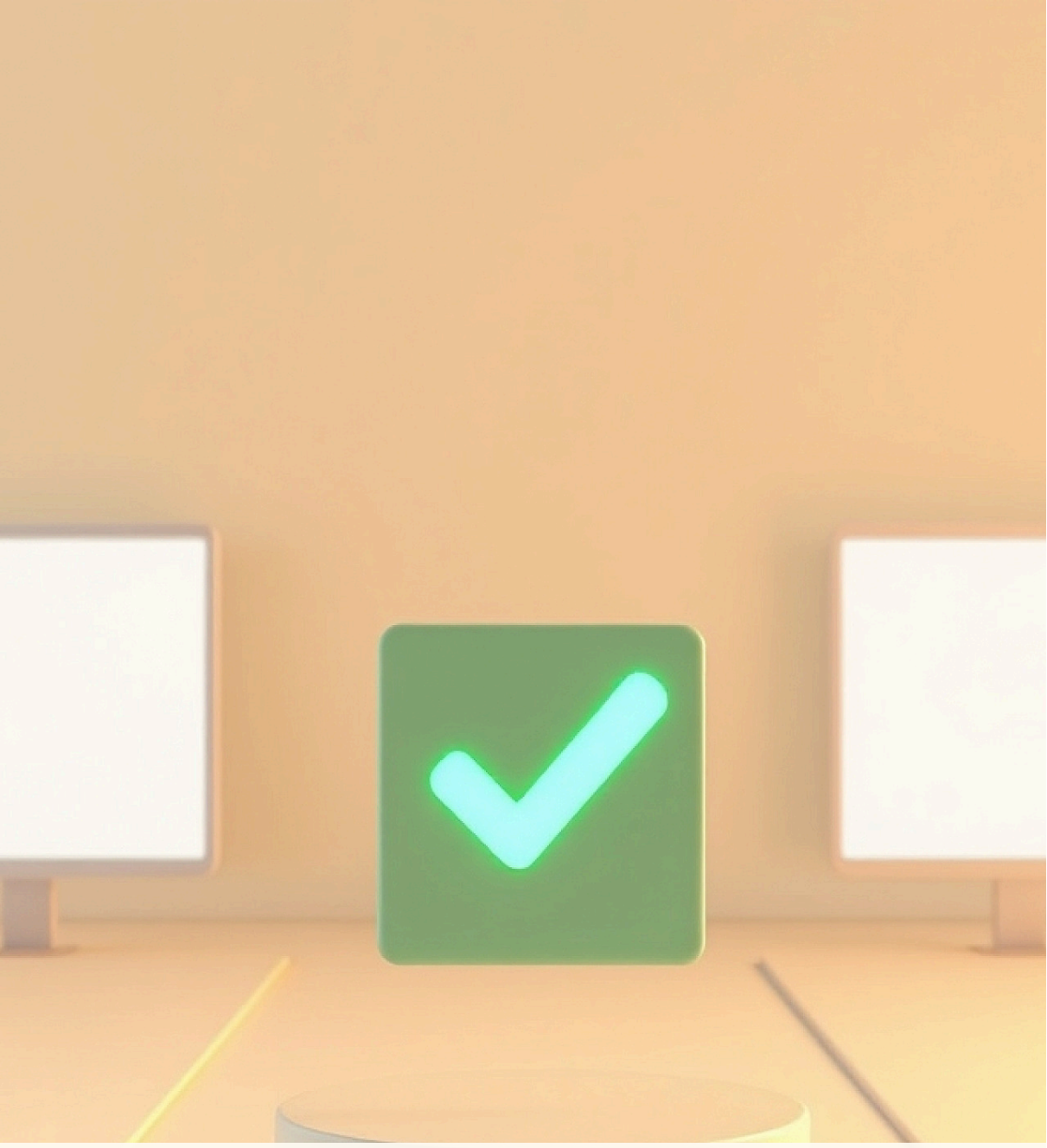
Combined conditions still produce a single true or false result. The if structure then acts on that one result as before. You do not need a special structure for combined conditions; the if works the same way. The program evaluates the whole combination and gets one answer. More detail does not change the basic idea.



A common beginner mistake is acting without checking the condition first. The program must evaluate the condition before it chooses an action. If you skip the check, the program cannot know which path to take. Always write the condition and let the if structure use its result. Checking first keeps the program's behavior correct and predictable.



Another mistake is expecting a condition to give something other than true or false. Keep in mind that a condition always evaluates to one of these two values. It will not return a number or a word as its result. If you need more detail, combine conditions rather than expecting a different kind of answer. True or false is always the outcome.



A condition lets a program check a situation and choose what to do based on a true or false result. If structures use conditions to make decisions, and several conditions can be combined for more detailed choices. Remember that the check always comes first, and the result is always true or false. With these ideas, you can read and write simple decisions in programs.