What is a logical value?

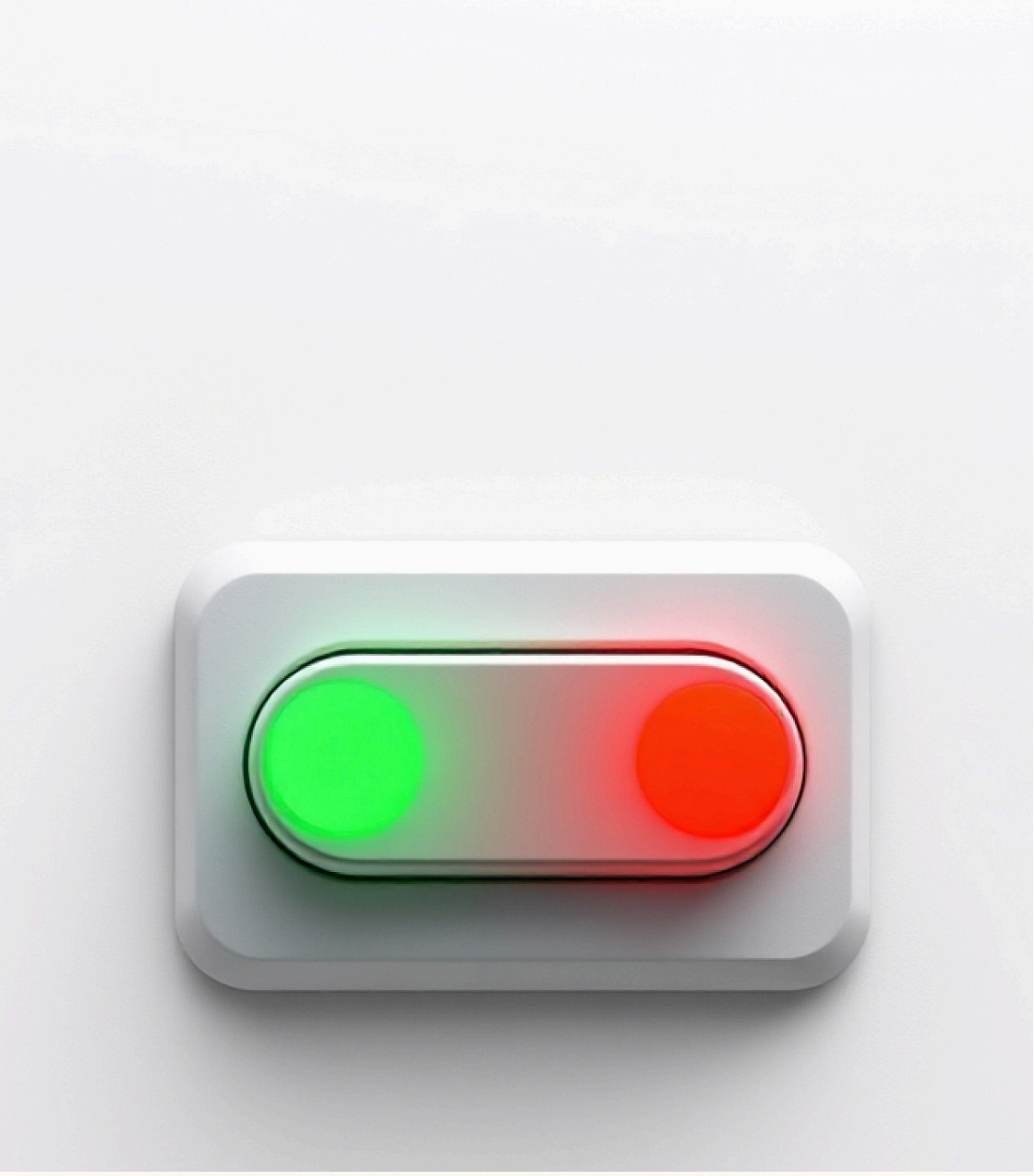


www.yzokul.com.tr

What is a logical value?



You meet the question: what is a logical value? The short answer is a value that is either true or false. That is the whole idea, and everything else in this book builds on it. A logical value is not a number and not a piece of text; it is a simple state with exactly two options. Keep this short answer in mind as you read the pages that follow.



A logical value has only two possible states: true and false. There is no third state, no middle ground, and nothing in between. You can think of it as a switch that is either on or off, but the only names you need are true and false. This two-state nature is what makes logical values so useful for clear decisions.



The two names used for these states are true and false. These words are the whole vocabulary of logical values. When you see true, it means the condition holds; when you see false, it means the condition does not hold. Learn these two words well, because every example in this book uses only them.



Look at a comparison like score > 50. The symbol > means greater than, so this asks a yes-or-no question: is the score greater than 50? You understand it as a question about two numbers. The comparison does not give a number back; it asks whether a certain relation is true. That question is the starting point for producing a logical value.



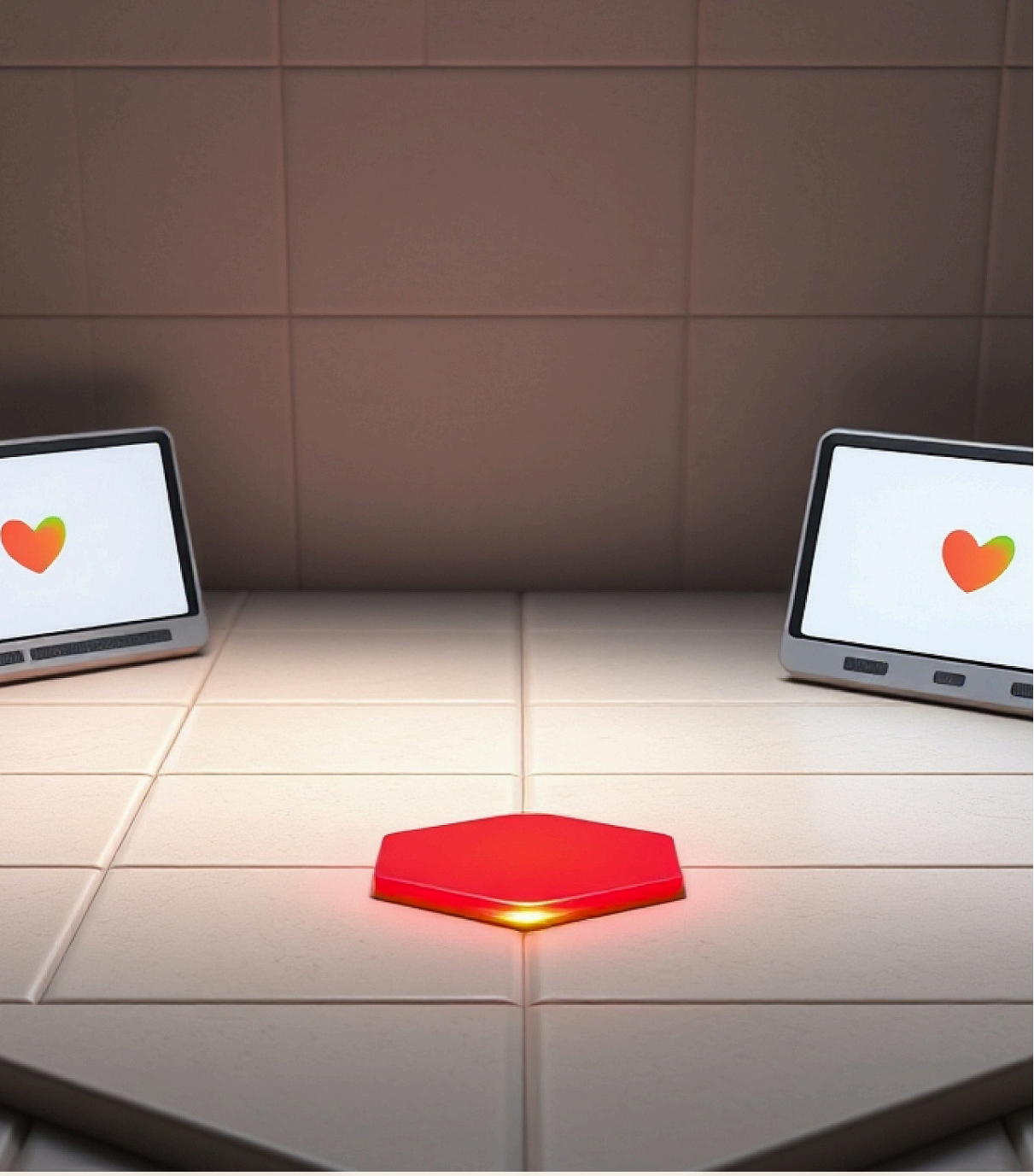
The comparison produces a logical value as its result. If score is 70, then 70 > 50 is true. If score is 30, then 30 > 50 is false. The comparison itself is the question, and the logical value is the answer it gives. This result is what a program can then use to decide what to do next.



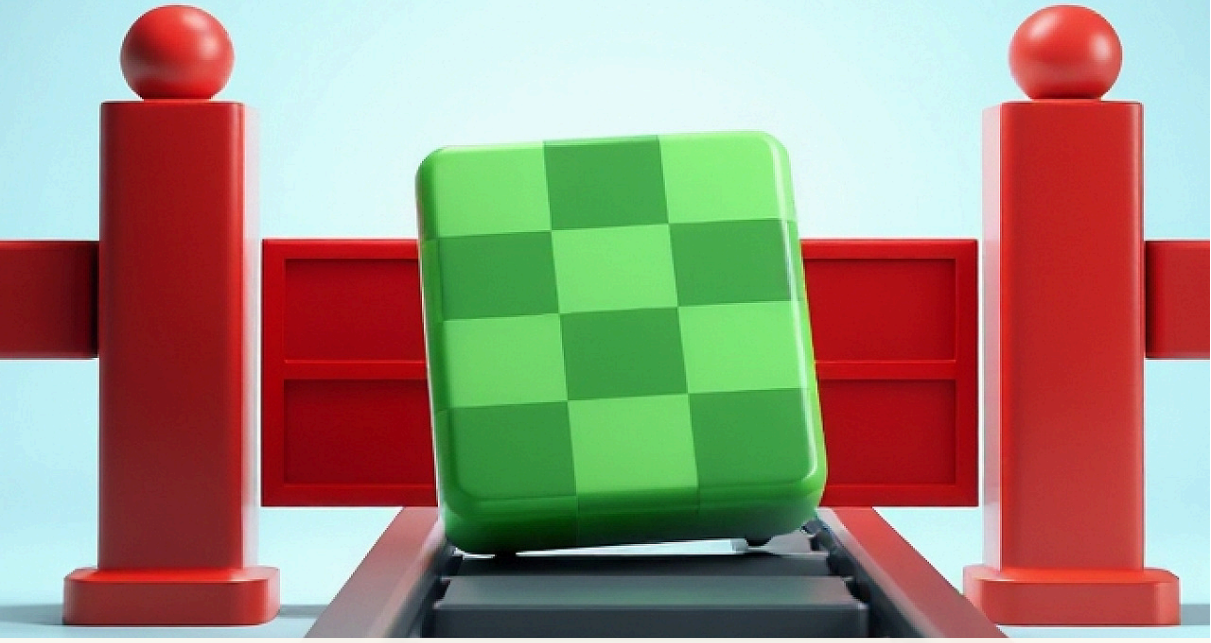
Compare two concrete cases side by side. With score = 70, the comparison score > 50 gives true. With score = 30, the same comparison gives false. Nothing else changes: the question is identical, but the answer differs because the score differs. This shows how one comparison can yield either of the two logical values.



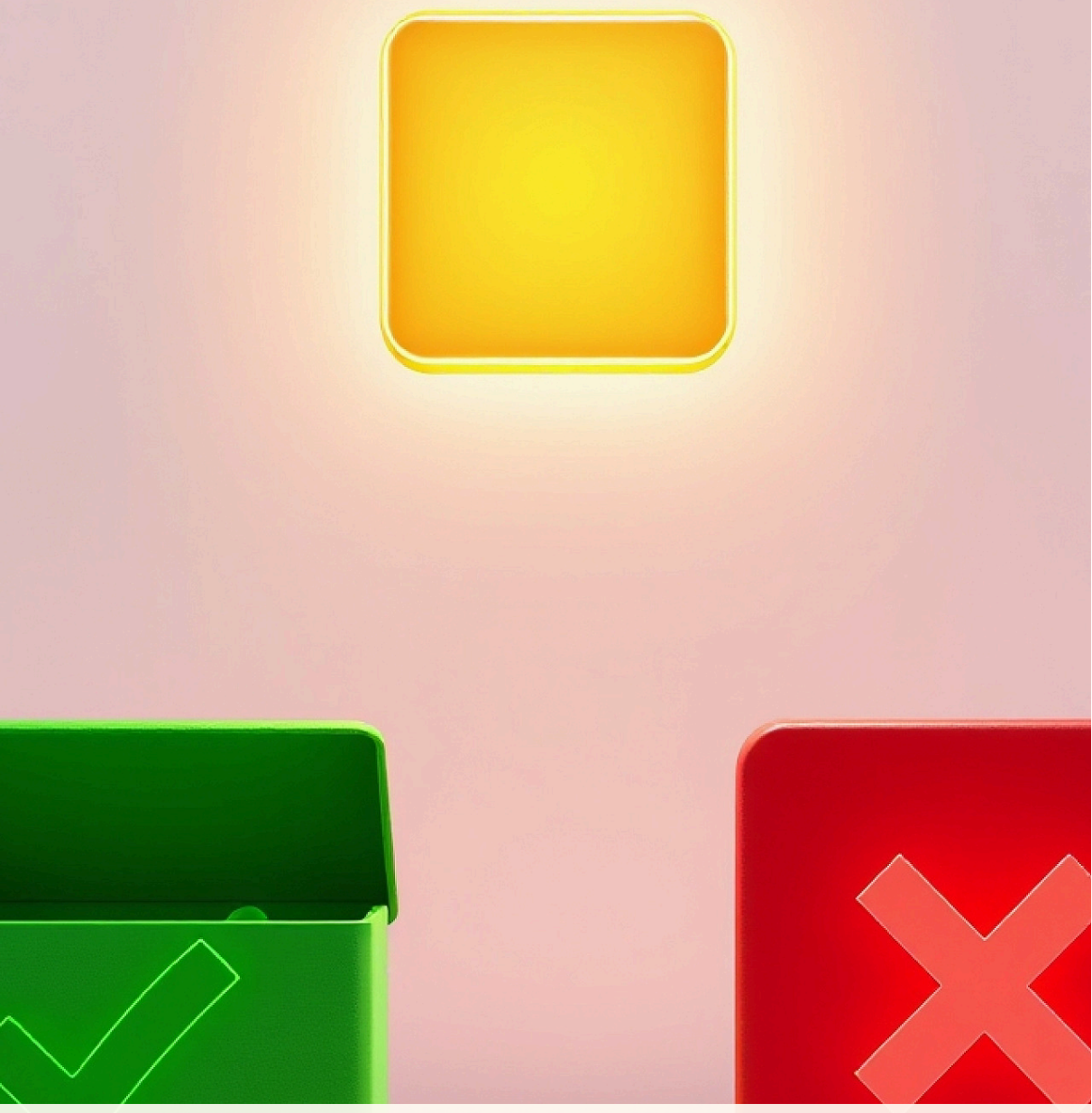
Programs use logical values when they must decide something. A true-or-false result steers what happens next in the program. Without logical values, a program would have no simple way to choose between alternatives. Decision making in programs depends on these two-state results, because each decision needs a clear yes or no.



At a decision point, the program checks a logical value. If the value is true, the program does one thing; if the value is false, it does another. The logical value selects the path. You can picture it as a fork in the road where the true-or-false answer tells the program which way to go. Only one path is taken each time.



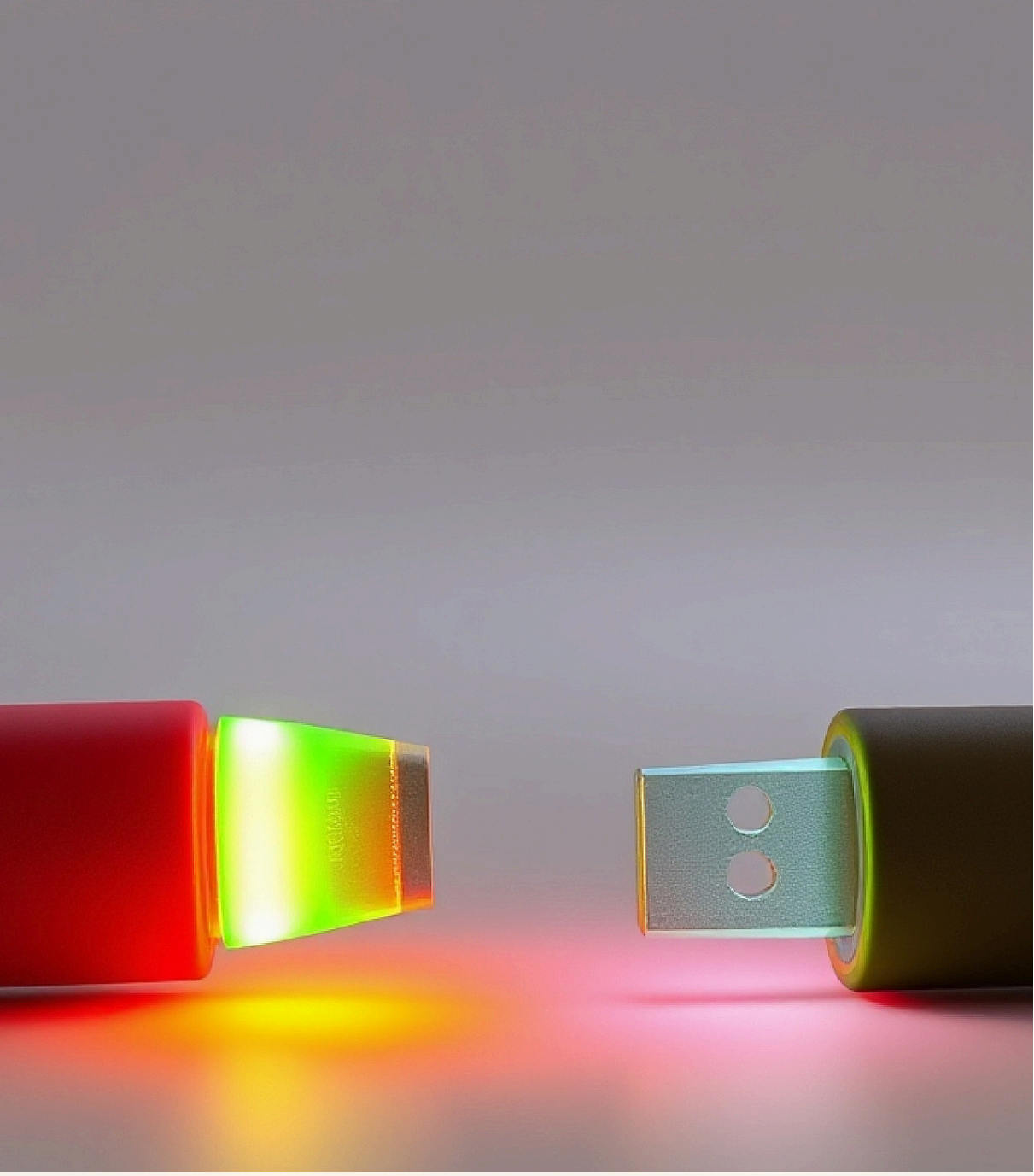
Trace a single example from comparison to chosen action. Suppose the program computes score > 50 and gets true. Because the value is true, the program runs the operation for a passing result and skips the other operation. The logical value is the reason one operation runs and the other does not. The comparison asked the question; the logical value answered it.



Apply this to an exam result. The comparison score > 50 being true can mean pass, and being false can mean fail. The program does not need to know the score itself at the decision point; it only needs the logical value. That true-or-false result is enough to choose between showing a pass message and showing a fail message.



Another everyday program question is: does this file exist? The answer is again a logical value, true or false. If the file exists, the result is true; if it does not, the result is false. The program can then act on that answer, for example by opening the file only when the value is true. No number or text is needed for the decision.



Consider one more case: is a connection open? Again the answer is true or false, and the program acts on it. If the connection is open, the value is true and the program can send data; if it is closed, the value is false and the program must wait or reconnect. The logical value guides this choice just as it did before.



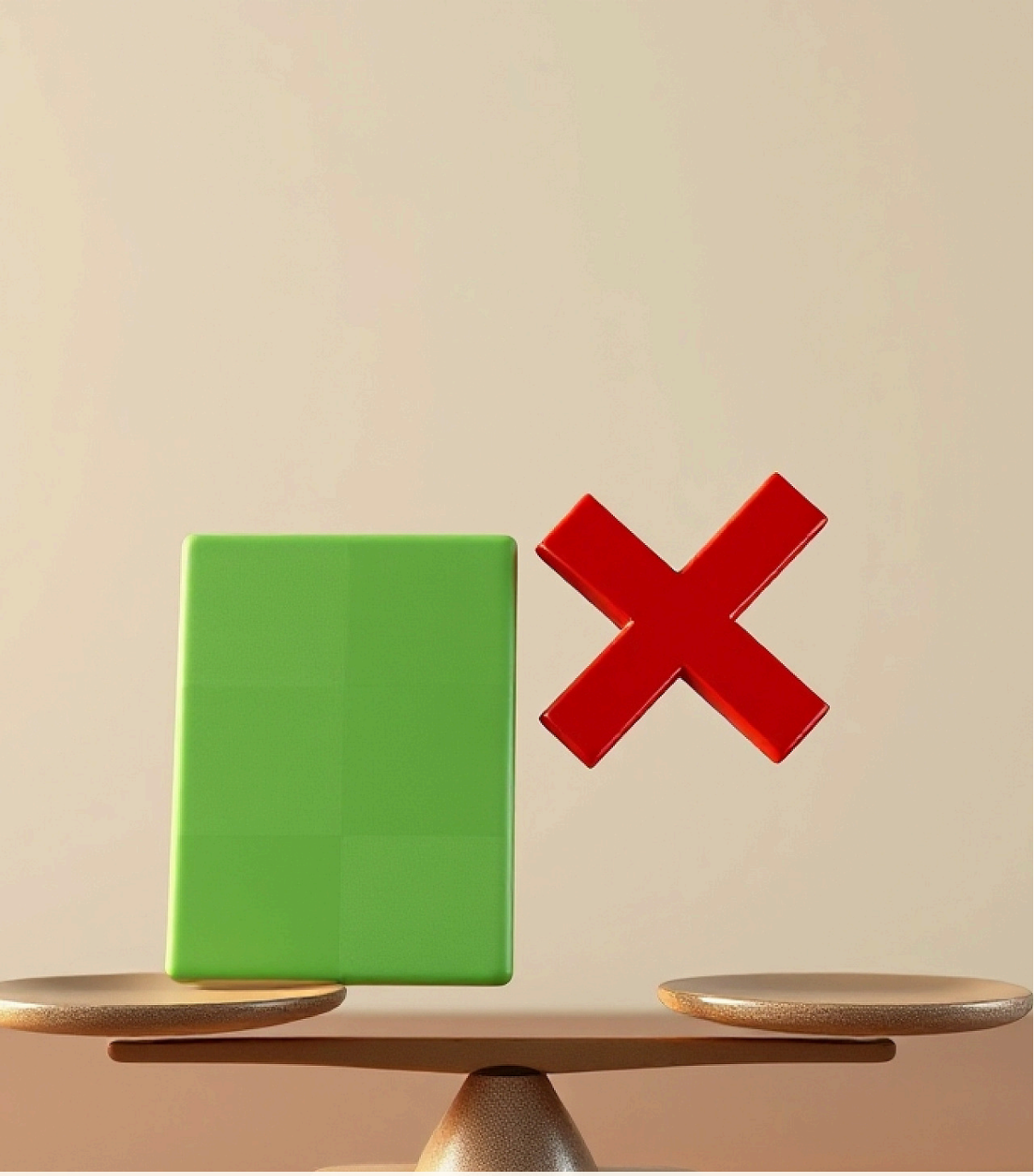
Notice that these examples share one pattern. A yes-or-no question produces a logical value that guides the program. Whether the question is about a score, a file, or a connection, the result is always true or false. Once you see this pattern, you can recognize logical values in many places inside program logic.



Check a common misunderstanding: a logical value is not a number or a text. It is only true or false. The number 1 is not the same as true, and the text true is not the same as the logical value true. Keep the two states separate from other kinds of values, because mixing them up leads to confusion when reading or writing programs.



Separate the comparison from its result. The comparison asks the question; the logical value is the answer it produces. For example, score > 50 is the question, and true or false is the answer. Do not call the comparison itself a logical value. The comparison is an operation, and its output is the logical value you use for decisions.



Sum up the answer: a logical value is true or false, produced by yes-or-no questions and used by programs to decide the next step. Remember the two states and their role in decisions. Whenever a program needs to choose between alternatives, it relies on a logical value. With this idea, you can follow how programs make simple decisions.