How are multiple conditions checked together?



www.yzokul.com.tr

How are multiple conditions checked together?



You want to know how multiple conditions are checked together. The short answer is that logical connectives like and, or, and not combine conditions, and parentheses control the order. A condition is a simple yes-or-no test. Logical connectives are words that join conditions. Parentheses are symbols that group parts of a rule.



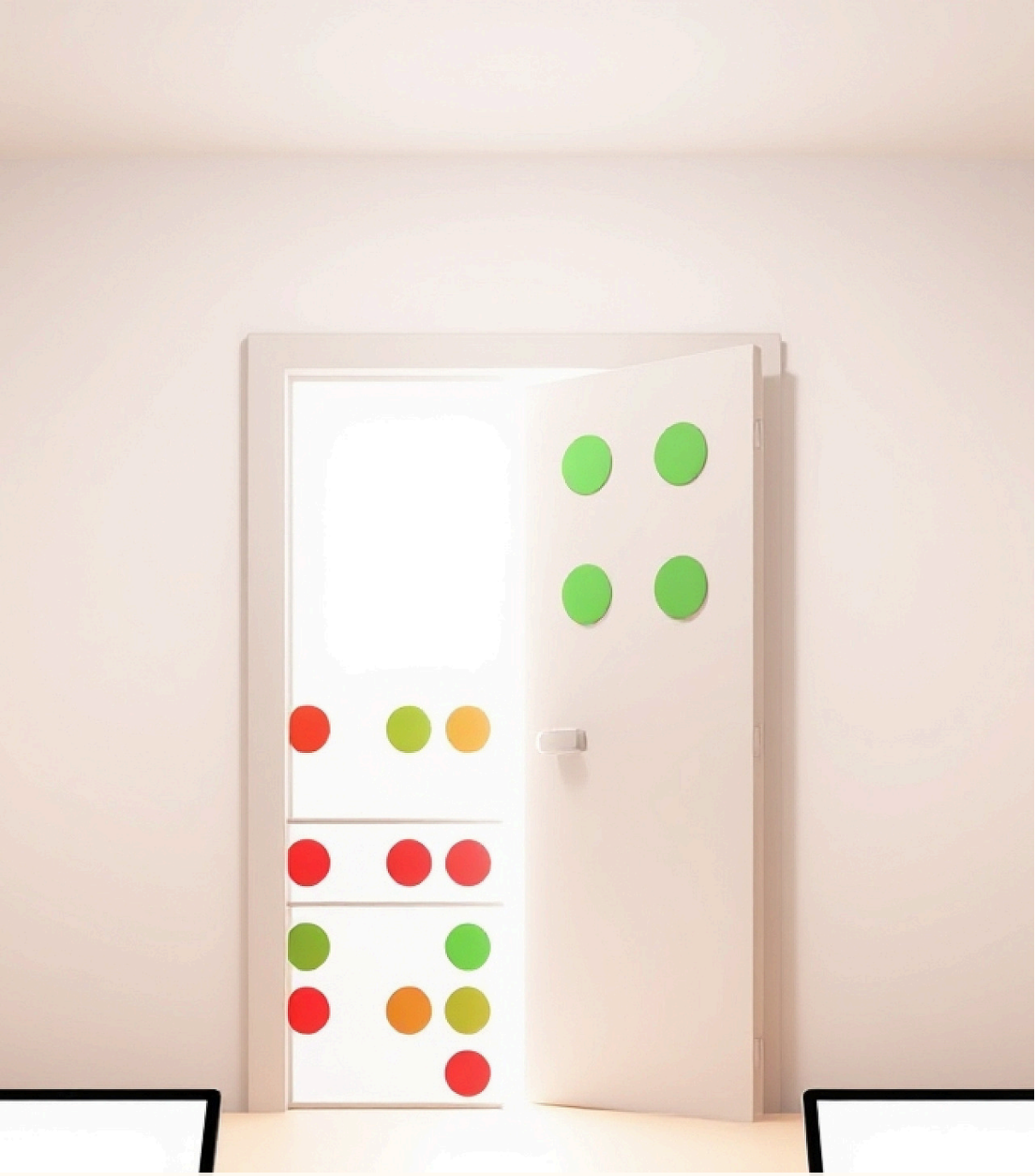
A condition is a simple yes-or-no test, like asking if the card is correct. Each condition gives either true or false, nothing in between. For example, the test of whether the card is correct returns true if the card matches and false if it does not. You can think of a condition as a question with only two possible answers.



Picture a door that needs a card and a password to open. Now there are two yes-or-no tests: is the card correct, and is the password correct? You must combine these two tests into one decision: should the door open? Combining conditions means deciding how their true or false results work together to produce a final true or false.



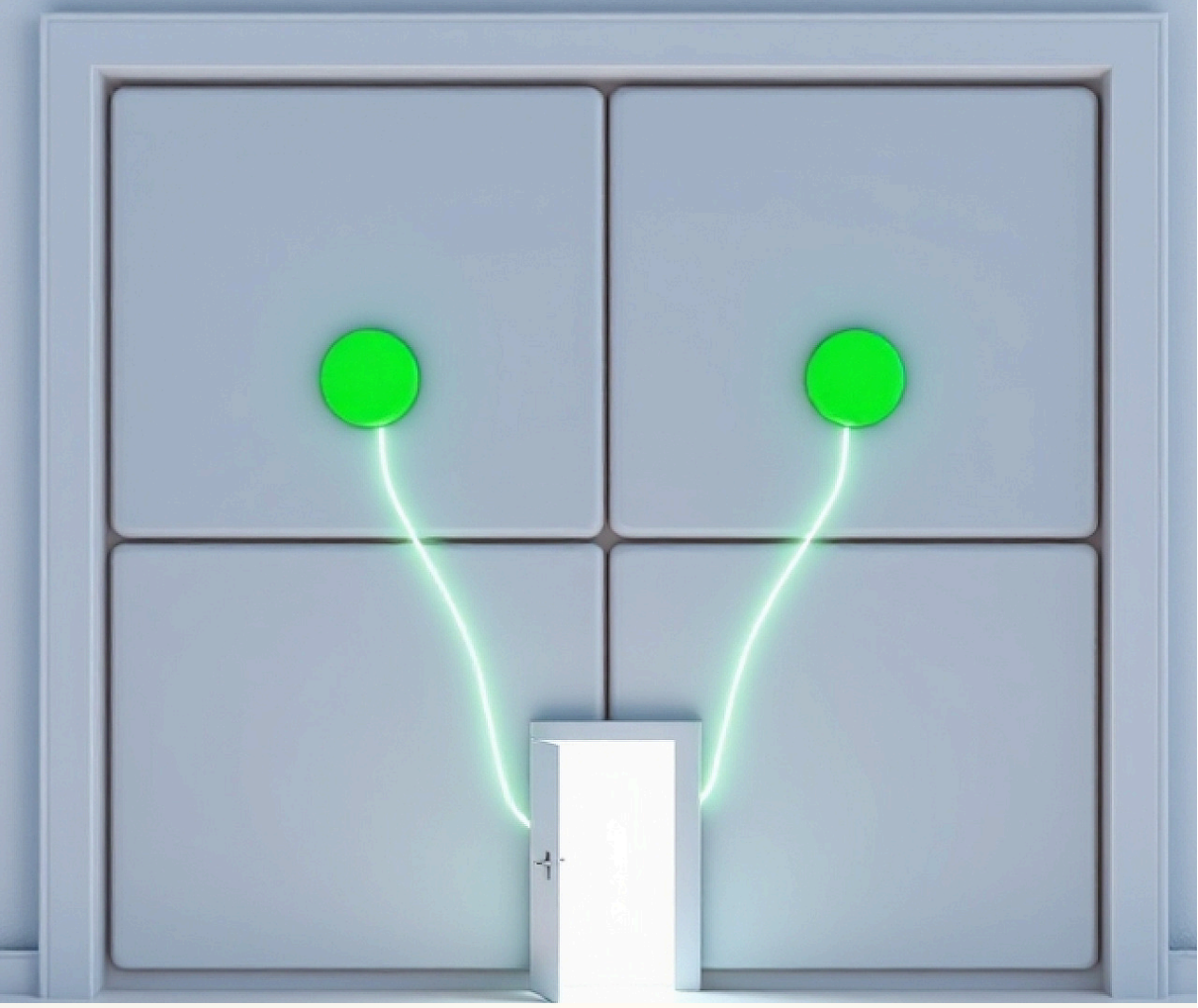
The and connective means every condition must be true for the whole check to be true. If the card is right but the password is wrong, then the whole check is false and the door stays locked. For example, card correct and password correct only passes when both are true. If either one is false, the result is false.



List all four combinations of card and password results: both true, card true and password false, card false and password true, both false. With and, only the row where both are true opens the door. The other three rows fail. This list is called a truth table, and it shows that and requires all conditions to be true.



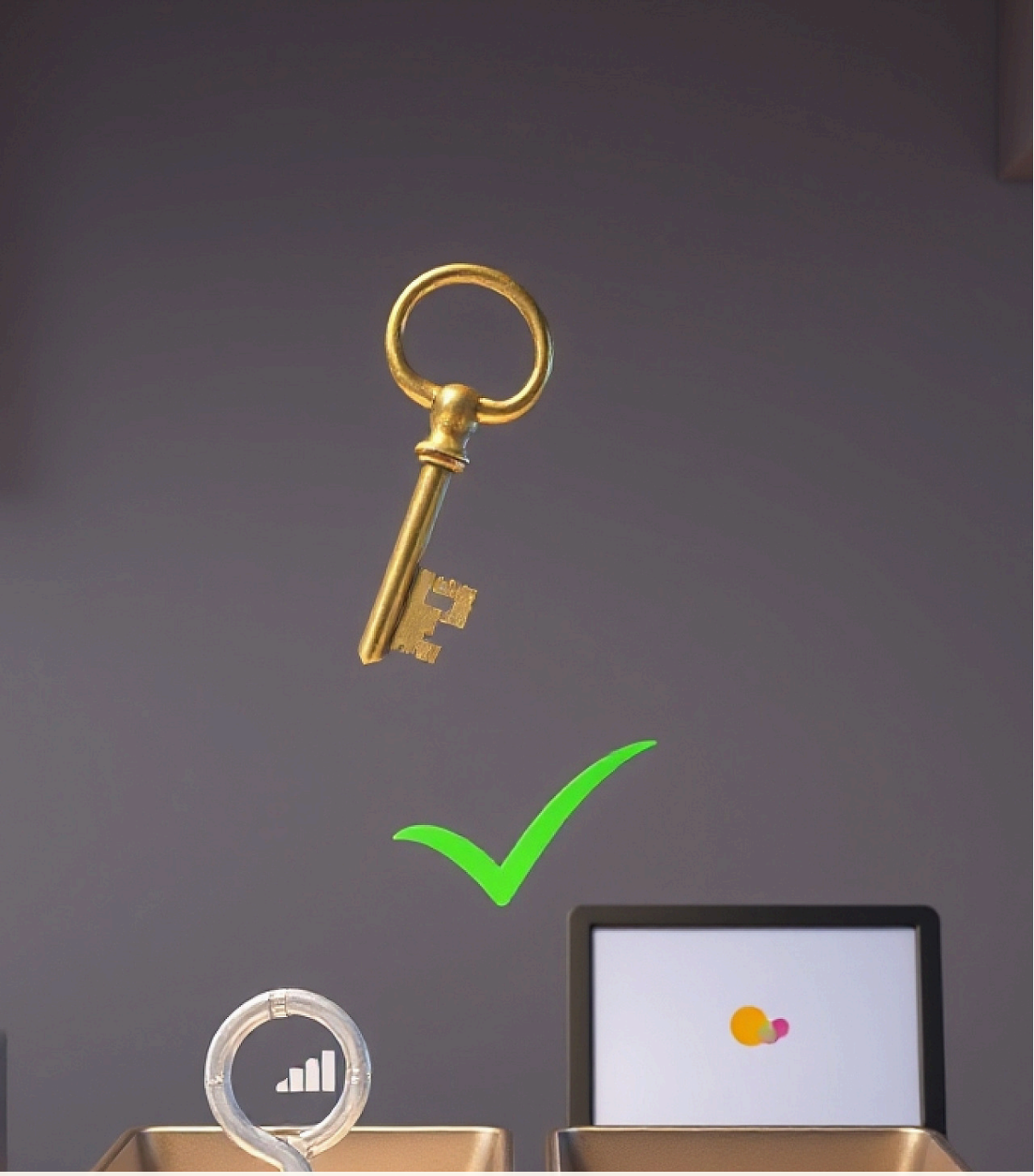
The or connective means at least one condition must be true for the whole check to be true. If either the card or the password is correct, the check passes and the door opens. For example, card correct or password correct passes when one or both are true. Only when both are false does the whole check fail.



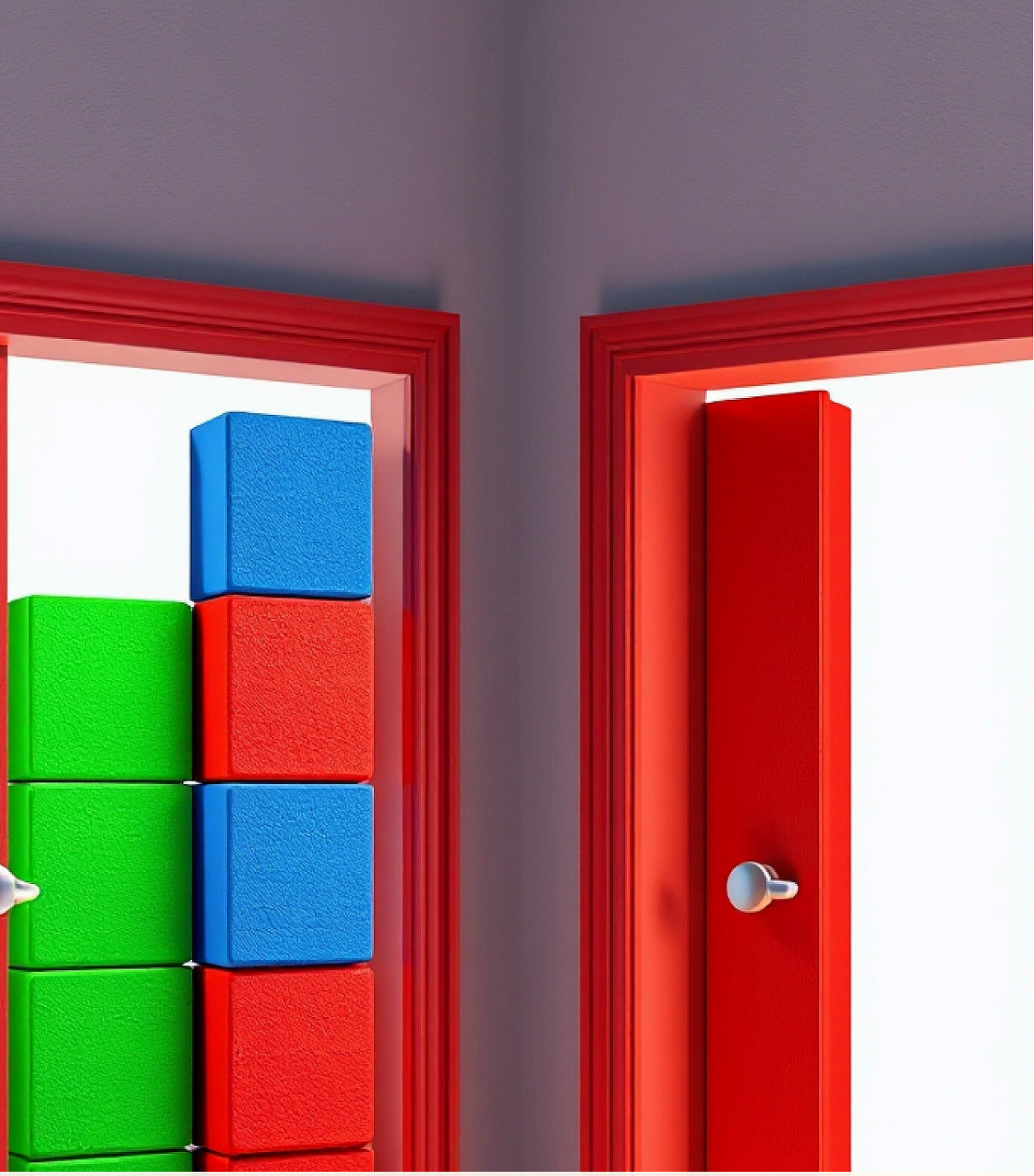
List the same four combinations again: both true, card true and password false, card false and password true, both false. With or, three rows pass: both true, card true only, and password true only. Only the row where both are false fails. This truth table shows that or needs at least one true condition.



The not operation flips a single result: true becomes false and false becomes true. It works on one condition, not on a pair. For example, not card correct is true when the card is wrong and false when the card is right. You apply not directly to one condition before combining it with others.



Read a rule like card correct and not password correct. Check the card first, then flip the password result with not, then combine with and. For example, if the card is correct and the password is wrong, not password correct is true, so the whole rule is true. If the password is correct, not password correct is false, so the rule is false.



Compare card and password or key evaluated two different ways. If you check card and password first, then or key, you get one answer. If you check password or key first, then and card, you get a different answer. The two orders can give different results, so the order of checking must be fixed to avoid confusion.



Whatever is inside parentheses is checked first. Parentheses make the intended order clear and remove ambiguity. For example, card and (password or key) means check password or key first, then combine with card. Without parentheses, card and password or key could be read either way, leading to different results.



Work through a rule with parentheses, and, or, and not step by step. Suppose the rule is (card correct or key correct) and not password correct. First check card or key: if either is true, that part is true. Then check not password correct: if the password is wrong, that part is true. Finally, combine with and: both parts must be true for the door to open.



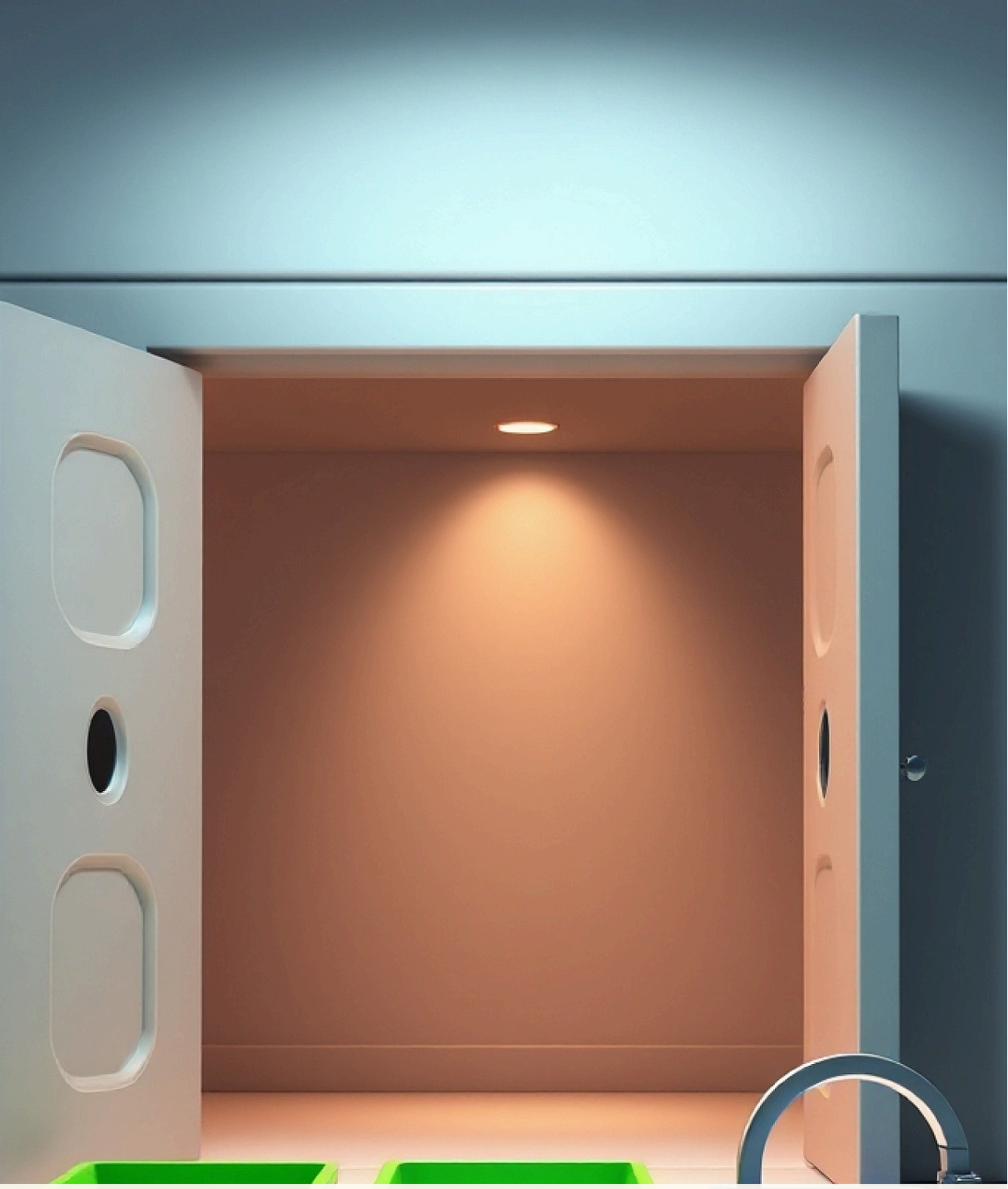
A beginner might use and when they mean or, so the door stays locked even when one condition is true. For example, if the rule is card correct and password correct but you only have the correct card, the door will not open. Check each connective against the real requirement: does the door need both, or just one?



A rule written without parentheses can be read the wrong way. For example, card and password or key might be read as card and (password or key) or as (card and password) or key. Fix it by adding parentheses around the part that should be checked first. Parentheses make the order clear and prevent mistakes.



Not applied to the wrong part of a rule flips the wrong result. For example, not (card correct and password correct) is true when either the card or the password is wrong, which is not the same as card correct and not password correct. Move not so it sits directly on the condition it should reverse, and use parentheses to be clear.



Remember these four ideas: and needs all conditions true, or needs at least one condition true, not reverses a single result, and parentheses set the order of checking. These four ideas cover checking multiple conditions together. Use them to build clear rules and to read complex conditions correctly.