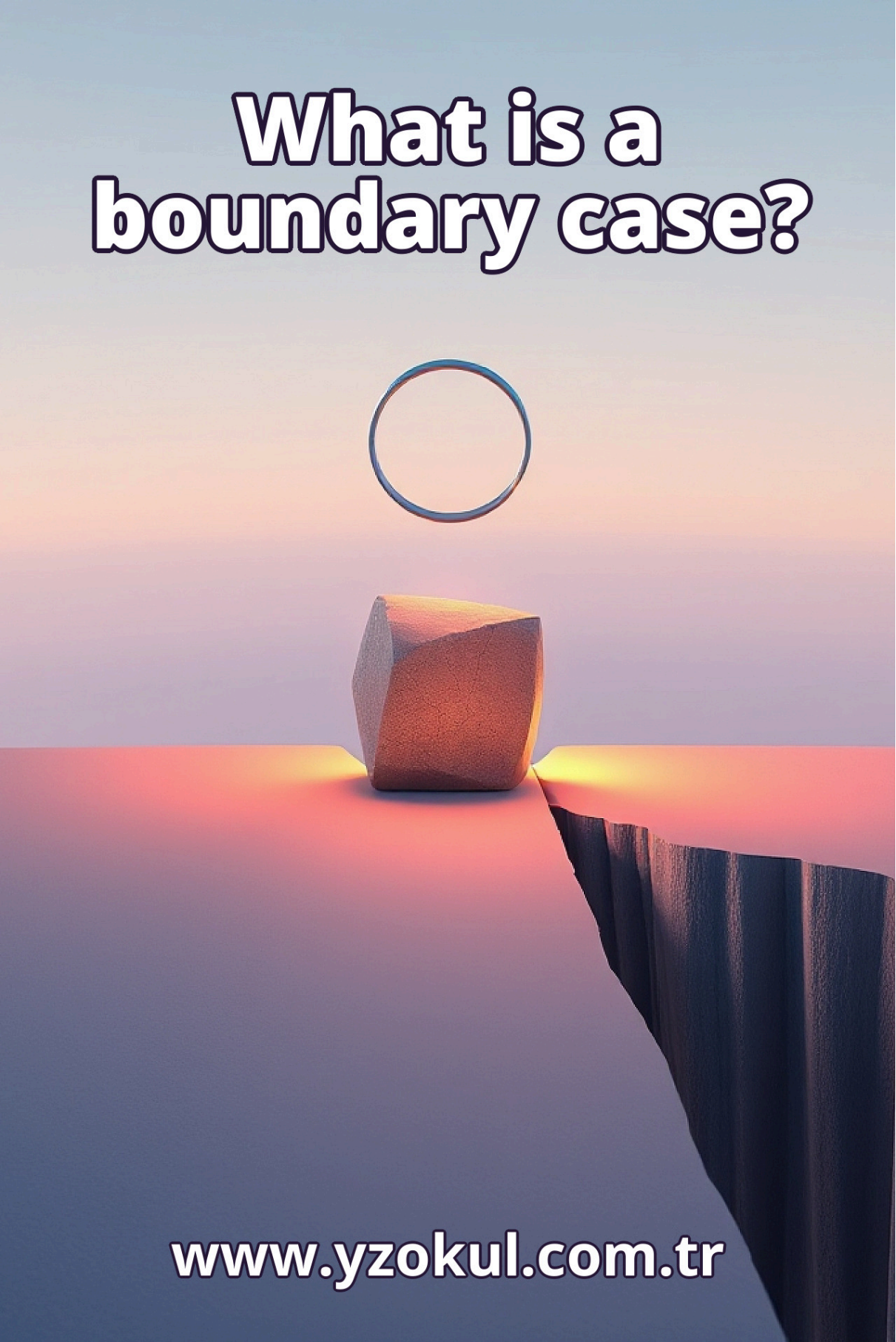
What is a boundary case?



www.yzokul.com.tr

What is a boundary case?



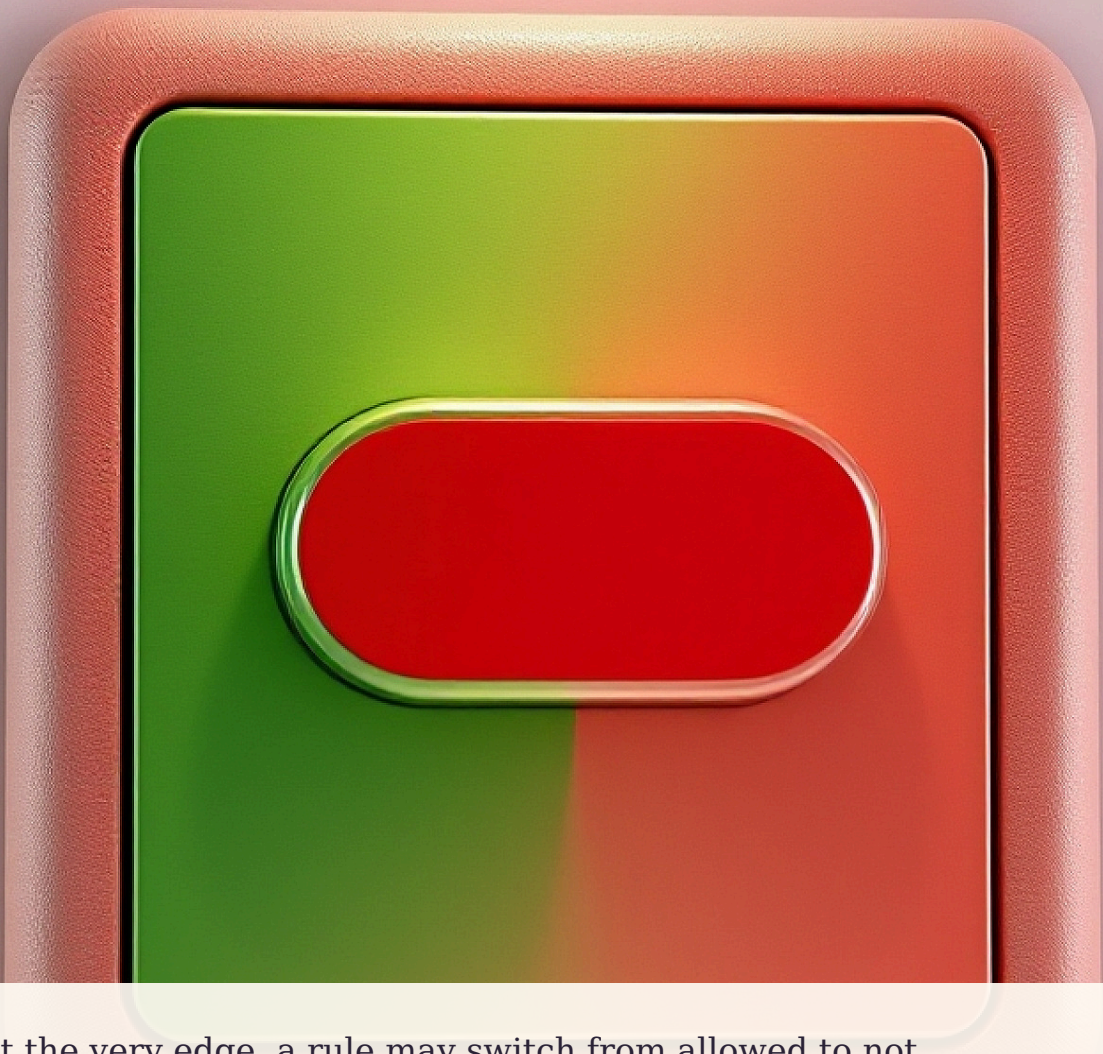
You ask a simple question: what is a boundary case? The short answer is that a boundary case is an extreme or special point where the rules of an operation change. An operation is any action a program performs, such as comparing numbers or reading a list. At most points nothing special happens, but at the edge the behaviour can switch. That edge point is the boundary case.



Start with a normal case, where everything works as expected. A normal case is an ordinary value in the middle of the allowed range, far from any edge. For example, a program that accepts ages from 0 to 120 handles the age 40 without trouble. This gives you a baseline, a known-good result to compare against later. First see the middle working, then look at the edges.



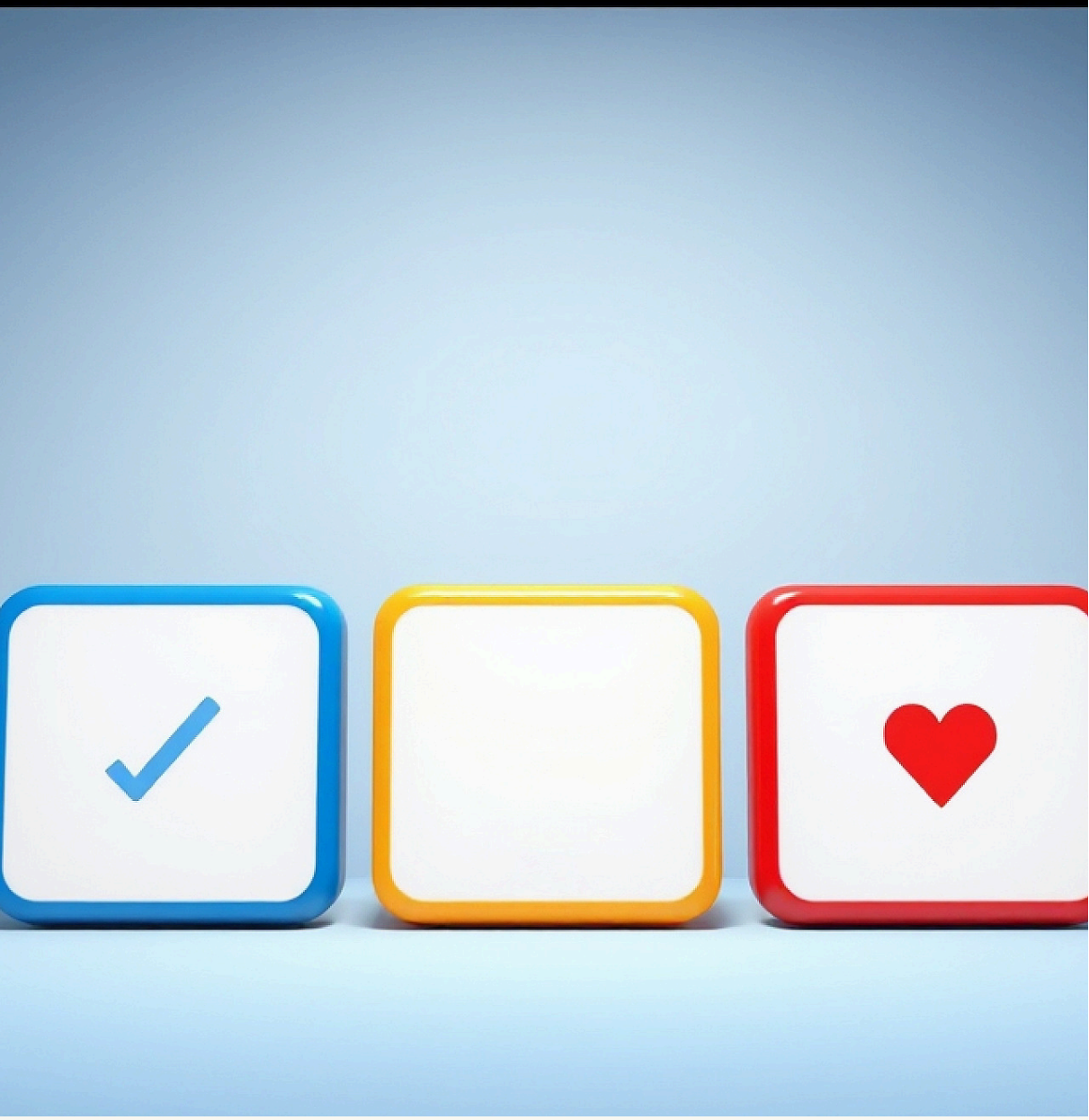
Picture a range of values with a start and an end, like the numbers 1 to 10. The edges are the first point and the last point of that range: here, 1 and 10. Everything between them is the inside of the range. A boundary case sits at one of these edges, or just outside them. The edge is where the range stops being ordinary.



At the very edge, a rule may switch from allowed to not allowed. Below the limit a value is accepted; above it, the value is rejected. The exact point where the switch happens is what makes a boundary case special. The rule does not change in the middle of the range, only at the edge. So the edge is the place worth watching closely.



Take a concrete rule: an age limit of 12. This means people aged 12 or younger are allowed, and people older than 12 are not. The single number 12 is the point where the rule changes from allowed to not allowed. Everything far below 12 is clearly allowed, and everything far above is clearly not. The interesting spot is right at 12.



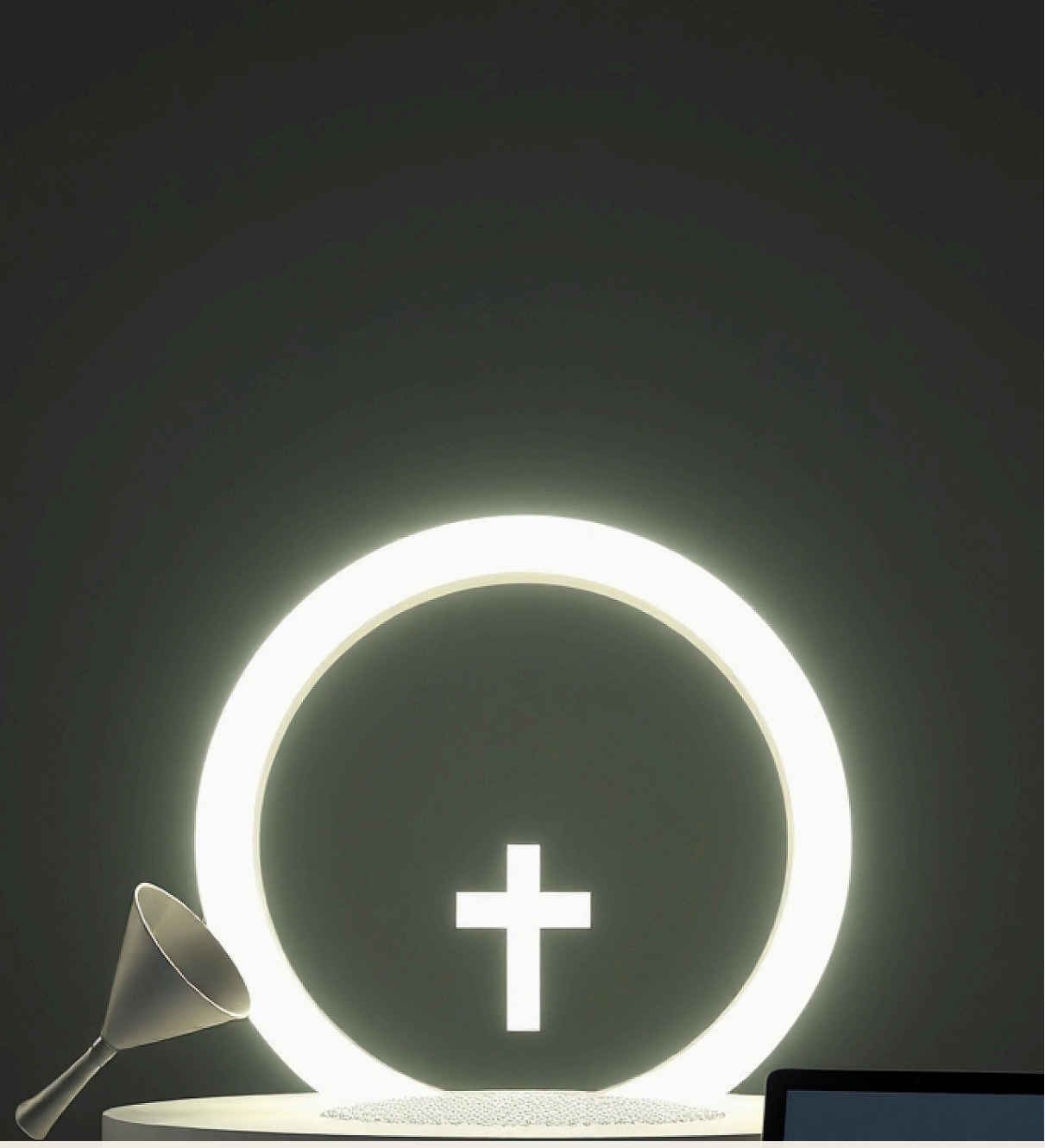
Try the three values around the limit: 11, 12 and 13. The value 11 is just below the limit, 12 is exactly at the limit, and 13 is just above it. Each one tests a different side of the rule. If the rule says 12 or younger is allowed, then 11 and 12 should pass and 13 should fail. These three values together check the switching point.



Testing only the value 12 is not enough. A program could accept 12 but wrongly reject 11, or wrongly accept 13, and you would never notice. The values on each side show whether the rule flips in exactly the right place. That is why you test just below, exactly at, and just above. Three values cover the whole edge.



Consider a list with nothing in it, called an empty list. It is a special point because there are no items to work with. Many operations behave differently when there is nothing to process: counting items gives zero, and taking the first item is impossible. An empty list is therefore a boundary case. Always try it as one of your test values.



Zero is another special value. Many operations behave differently at zero than at ordinary numbers. Dividing by zero is not allowed, and multiplying by zero wipes a value out completely. Counting from zero is common in programming, where the first position of a list is often numbered zero. Because behaviour changes there, zero counts as a boundary case.



Look at the top end of an allowed range. If a form accepts a number up to 100, then 100 is the largest allowed number, and it is a boundary case too. The value 99 sits safely inside, but 100 is the last point that still follows the rule. Testing the top edge checks that the program does not cut off too early or too late.



A program often works fine on ordinary, middle-of-the-range examples. Feed it a normal age like 40, and it answers correctly. Everything looks right at first, and you may feel confident. Normal examples pass again and again. But passing them does not prove the edges are handled well, because the edges were never tried.



The same program can fail at an extreme point. Perhaps it rejects the age 12 when it should accept it, or accepts 13 when it should not. The mistake hides during normal use and only appears at the boundary. This is common: the middle of the range is easy, and the edge is where careful logic is needed. That is why boundary cases deserve attention.



Testers give these points special attention because that is where errors are most likely to hide. The middle of a range is simple and rarely goes wrong. The edge demands exact comparisons, and small slips there are easy to make. Checking the boundary cases catches these slips early, before real users meet them. A few extra tests at the edges save a lot of trouble.



A common mistake is testing only normal examples. You try a few ordinary values, everything passes, and you conclude the program is correct. This gives false confidence, because the edges stay unchecked. The program may still break at 12, at zero, or at an empty list. Always add boundary values to your normal tests, not instead of them.



Another common mistake is checking only one side of a limit. You test 12 and 13 but forget 11, or you test the empty list but forget the list with one item. The rule can fail on either side, so one side alone proves little. You need the values just below and just above the limit. Test both sides, and the exact point between them.



Sum up the answer: a boundary case is an extreme or special point where the rules of an operation change. Remember the practical habits: test 11, 12 and 13 around a limit, try an empty list, try zero, and try the largest allowed number. Programs often work on normal examples but fail at these points. Give the boundaries special attention in your tests.