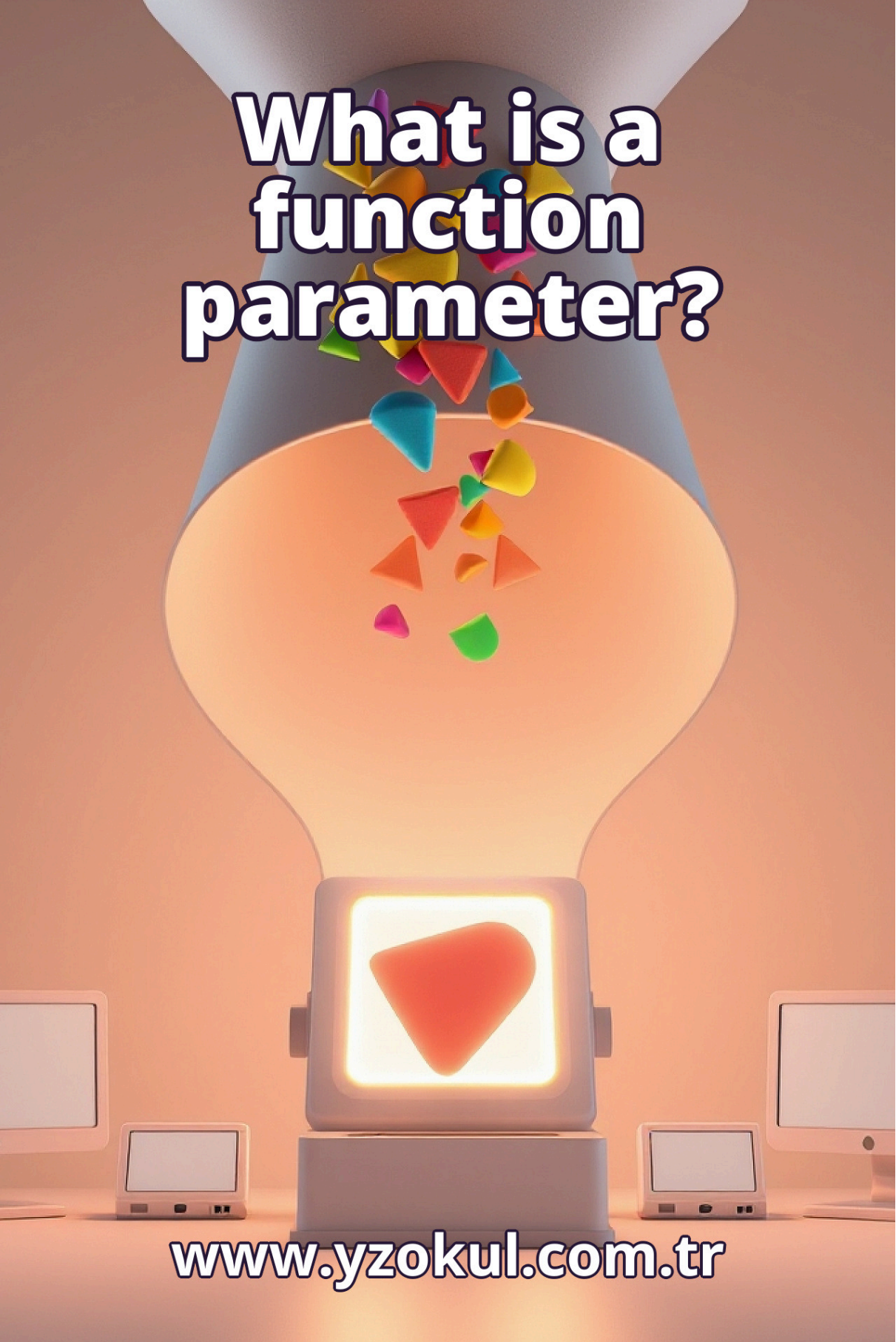

What is a function parameter?
www.yzokul.com.tr

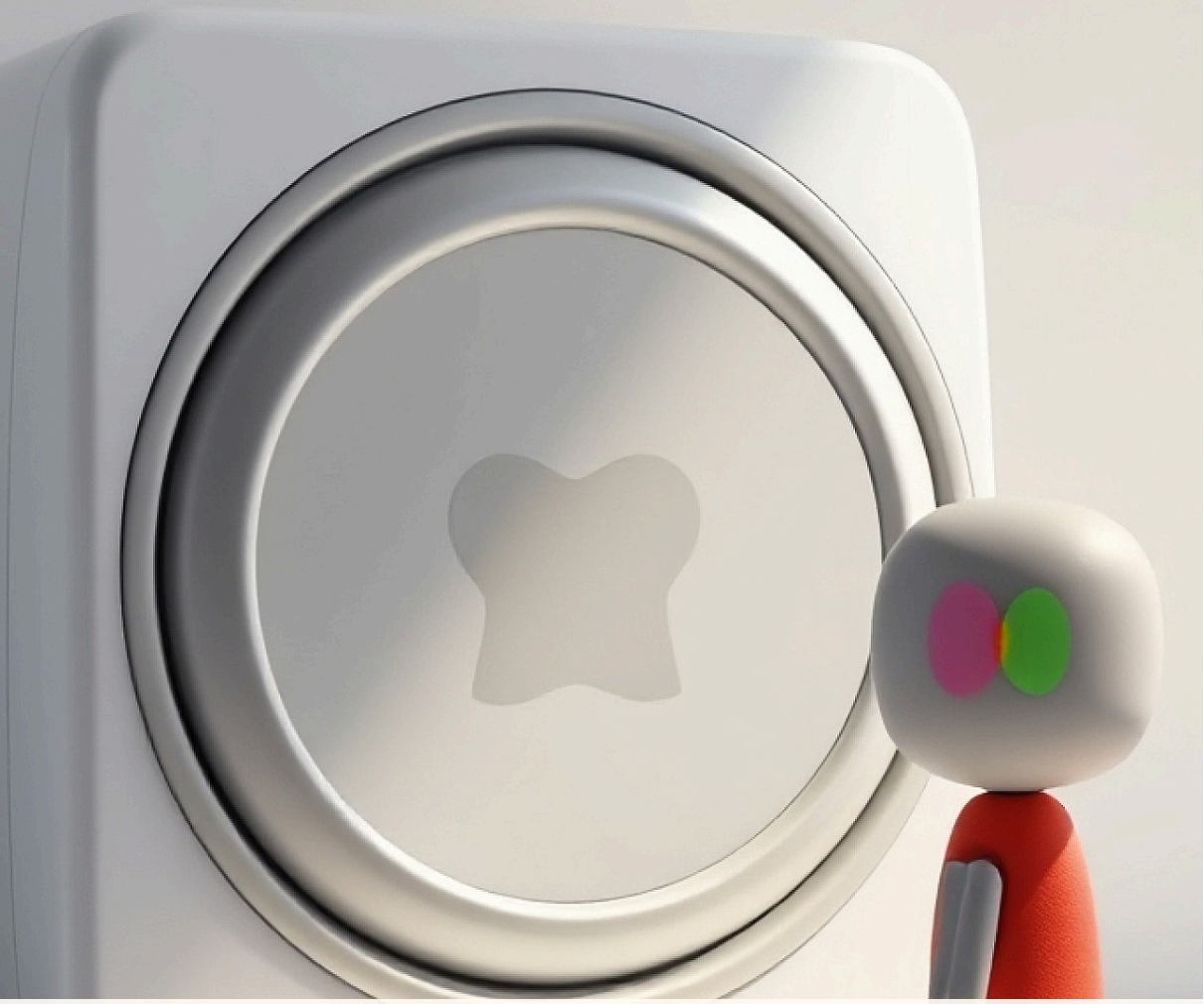
What is a function parameter?



A function parameter is the name for information that a function receives from outside. Think of it as a labeled spot where the function expects a value. The short answer: a parameter is a name inside a function that stands for incoming information. This book explains that idea step by step, starting from zero, so you can see exactly how parameters work and why they matter.



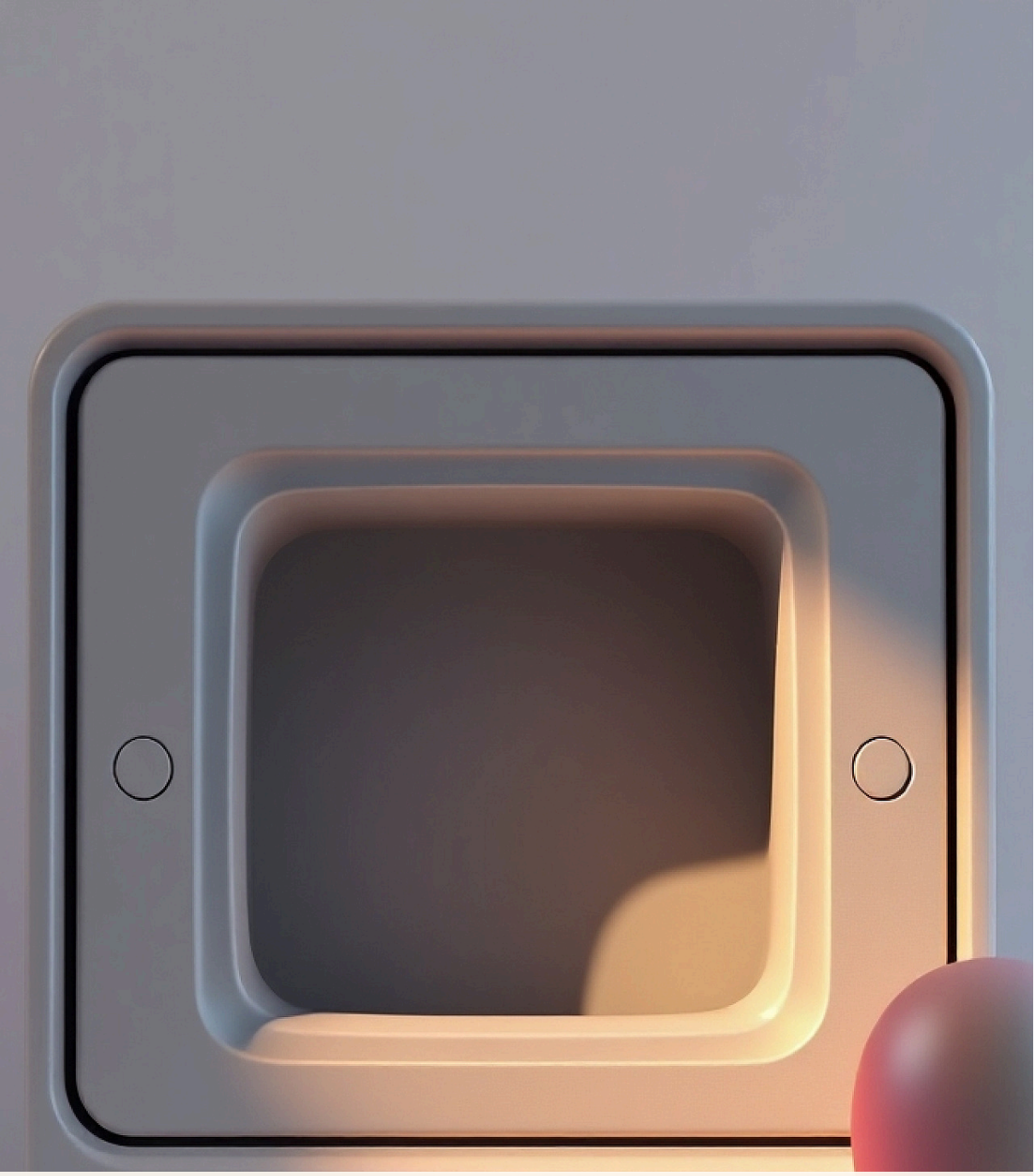
A function is a small named piece of work that does one job when you call it. You give it a name, and later you ask it to run by saying its name. For example, a function called greet could do the job of saying hello. Functions are reusable: you write the job once and run it many times. This saves effort and keeps your code organized.



Some jobs need to know something from outside to work well. Greeting a person is one example: to say hello properly, the function must know who the person is. Without that information, it can only give a generic greeting. The function cannot do its job well unless it receives that outside detail each time it runs. This is where parameters come in.



When a function runs, some information must come from outside it. That incoming information is not created inside the function; it is supplied each time the function is used. A parameter is the name for that incoming information. It is like a label the function uses to refer to whatever value arrives. The parameter itself is not the value; it is the name for it.



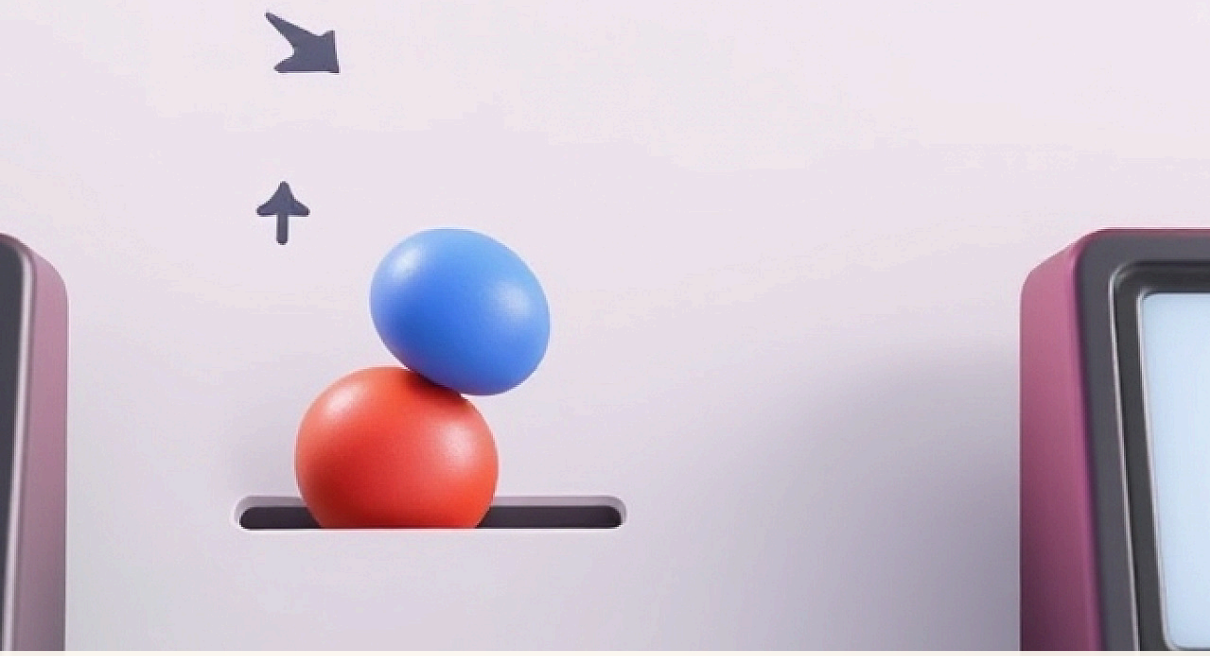
Here is a greeting function with a parameter named name. The parameter is just a label inside the function that stands for the information to come. It tells the function: when you run, you will receive something called name. The function can then use that name in its work. The parameter has no value yet; it only marks what will arrive.



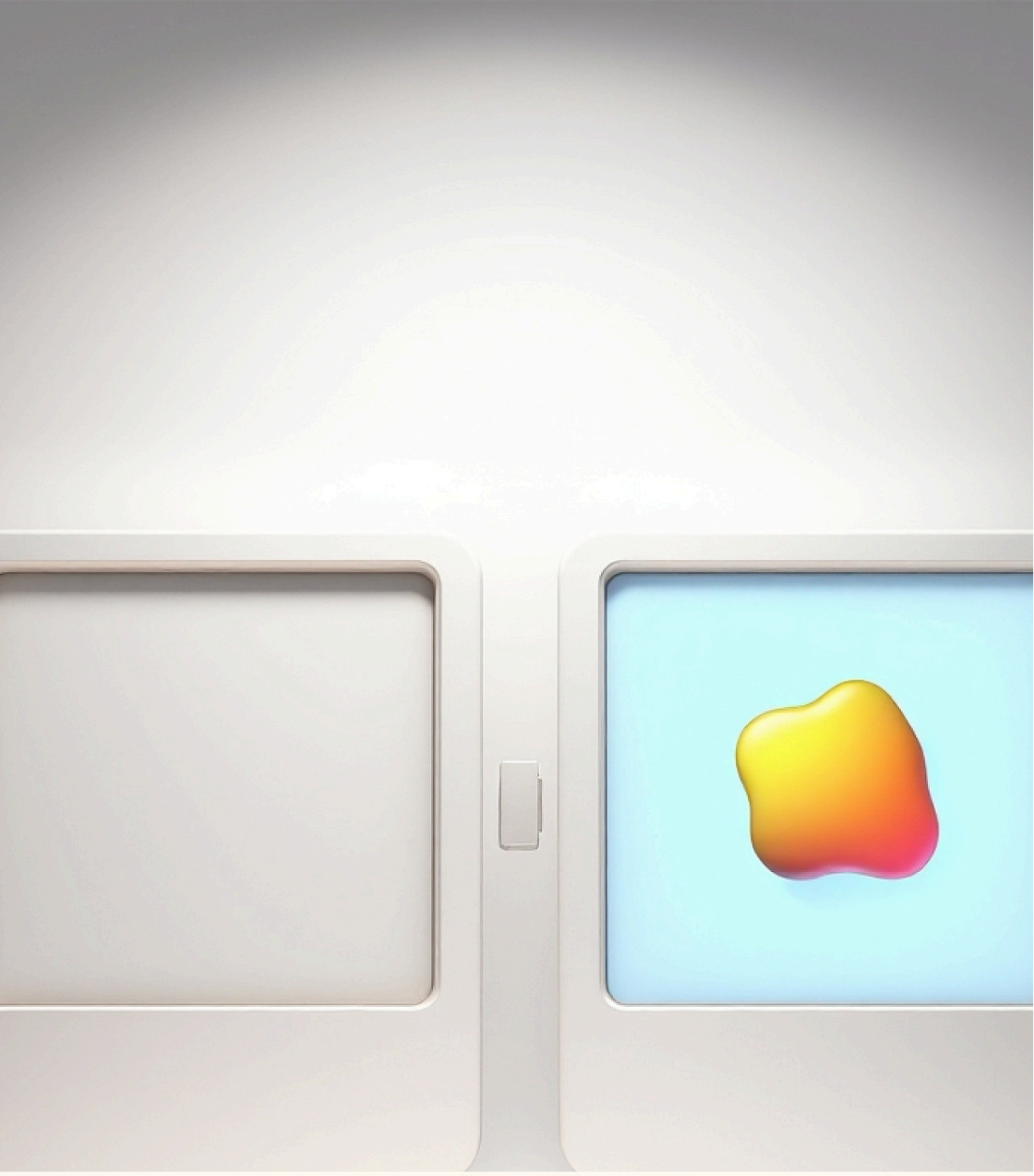
Picture the parameter as an empty slot inside the function that waits to be filled. The slot has a name, such as name, but no value yet. When the function is called, the slot gets filled with whatever information is sent. Until then, the slot stays empty. The parameter is the name of that slot, not the content.



Calling a function means asking it to run. You write the function's name and, in parentheses, the information you want to send. When it is called, the outside information is sent to fill the parameter slot. For example, calling greet with a person's name fills the name parameter. The function then runs with that information available inside.



The actual value sent when the function is called is called an argument. The parameter is the name; the argument is the real value. For example, if the function has a parameter called name and you call it with the value Maria, then Maria is the argument. The parameter name receives that argument. So the argument fills the parameter slot.



Compare the two words side by side. A parameter is the name written in the function definition, like name. An argument is the value you send when you call the function, like Maria. The parameter is inside the function; the argument is outside at the call. Remember: parameter is the name, argument is the value. This clears up the most common confusion.



The same greeting function can be used for different people by sending different arguments. Call it with Maria, and it greets Maria. Call it with Sam, and it greets Sam. One function works for many cases without being rewritten. You do not need a separate function for each person. The parameter makes the function flexible and reusable.



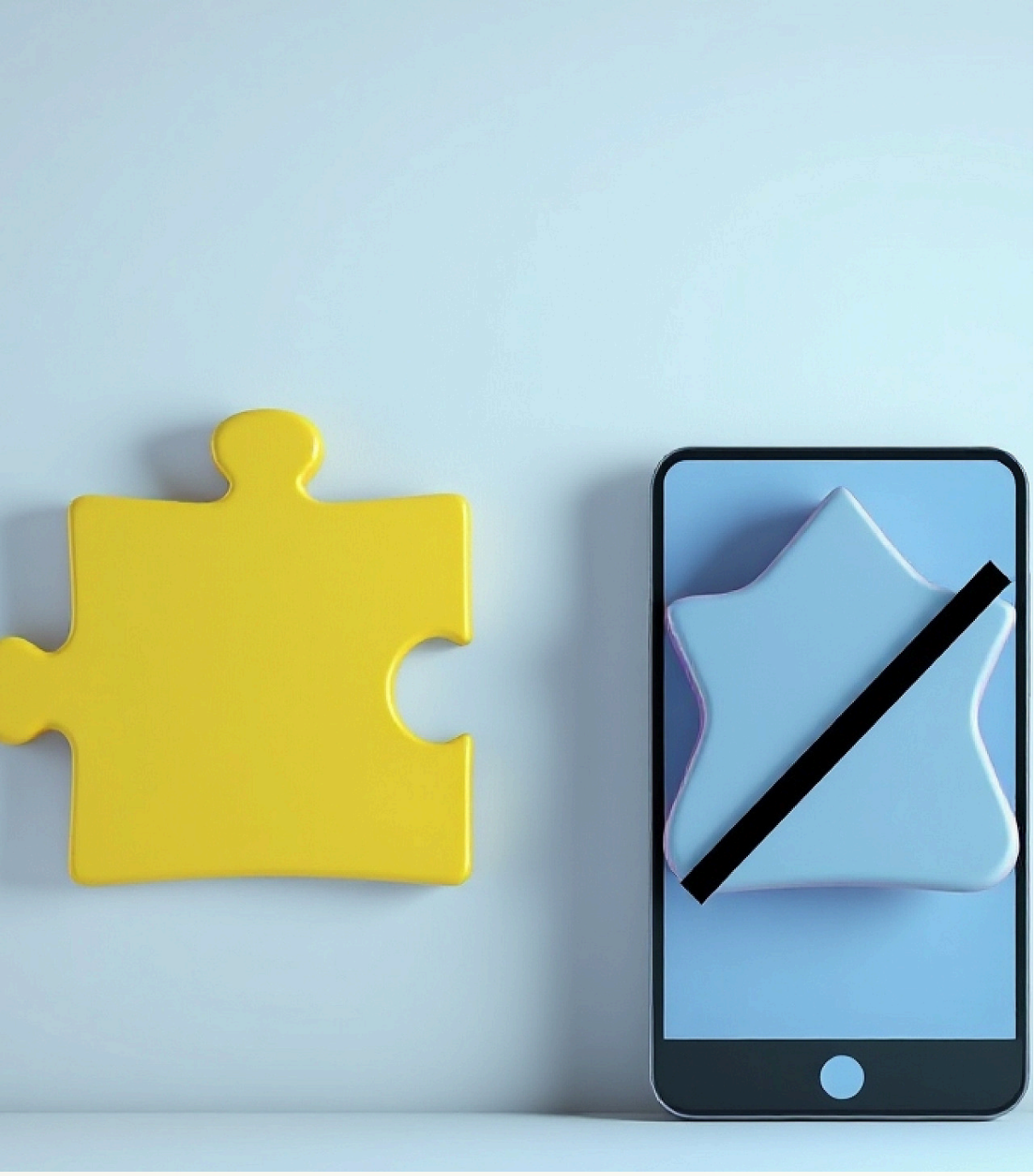
Without parameters, the function would use the same fixed value every time. It might always greet the same name, no matter who is there. Parameters stop the function from being stuck on one case. They let the function receive new information on each call. This is why parameters are essential for writing useful, general functions.



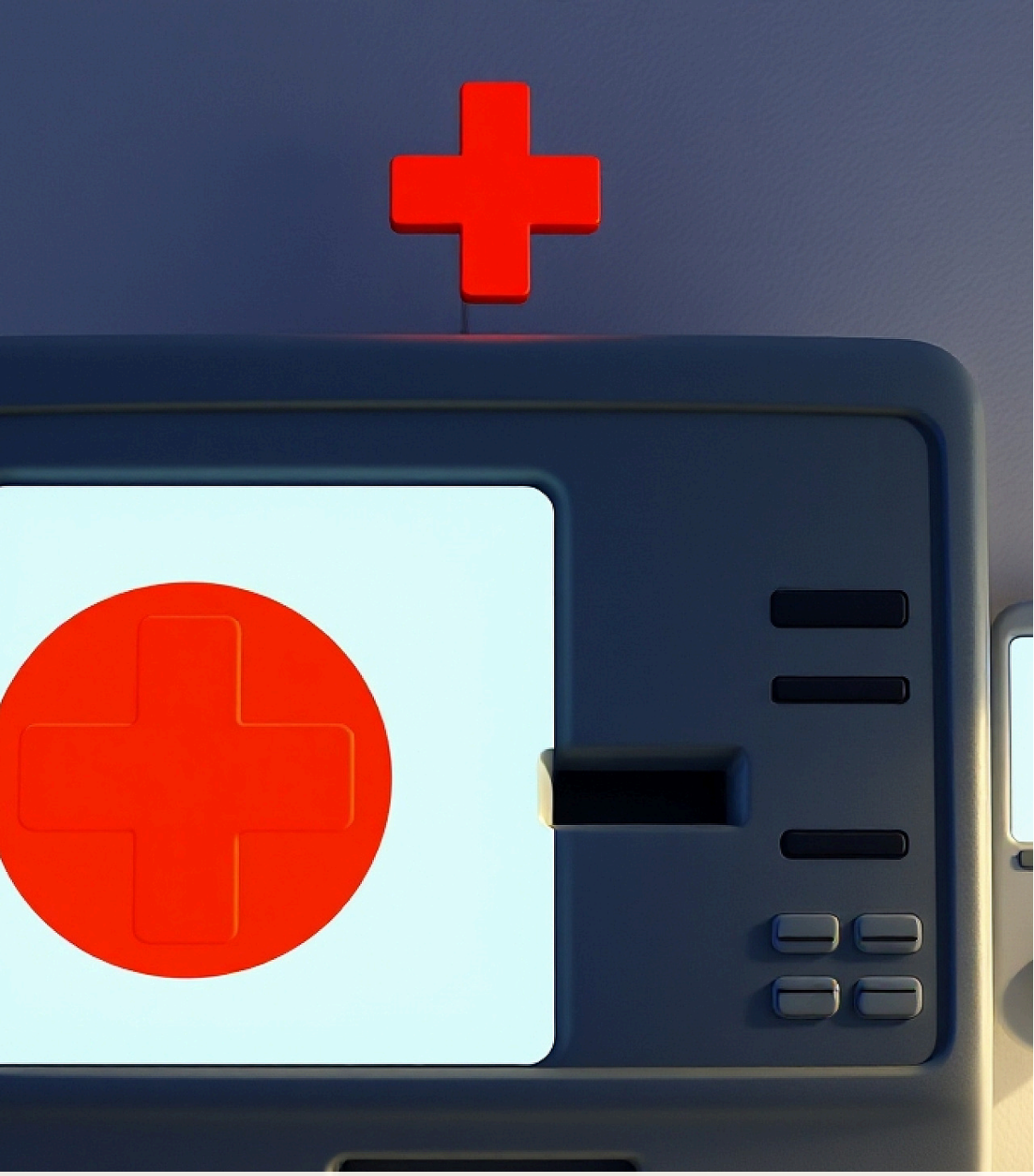
A function can have two parameters at once. For example, a greeting function might take a name and a time of day. Each parameter receives its own argument when the function is called. You send two values in the call, and each fills its matching parameter. This lets the function do a more specific job using both pieces of information.



Arguments are matched to parameters by their position. The first argument fills the first parameter, the second argument fills the second parameter, and so on. Sending values in the wrong order gives the wrong result. If you send the time of day where the name should go, the greeting will be wrong. Order matters when you call a function.



A common mistake is swapping the words: calling the value a parameter or the name an argument. The fix is to remember name inside, value outside. The parameter is the name written in the function. The argument is the value you send when calling it. Keep that rule in mind, and you will use the two words correctly.



What happens if a function expects a parameter but no argument is sent? The slot stays empty, so the function cannot do its job. Many programming languages report an error in this case. Always send the right number of arguments when you call a function. Check that every parameter gets a value, or the function will not work as intended.



A parameter is the name for information a function receives from outside. The value you send when calling the function is the argument. Parameters let one function work for many different cases without being rewritten. Remember: name inside, value outside. With this idea, you can understand and use functions that adapt to different inputs.