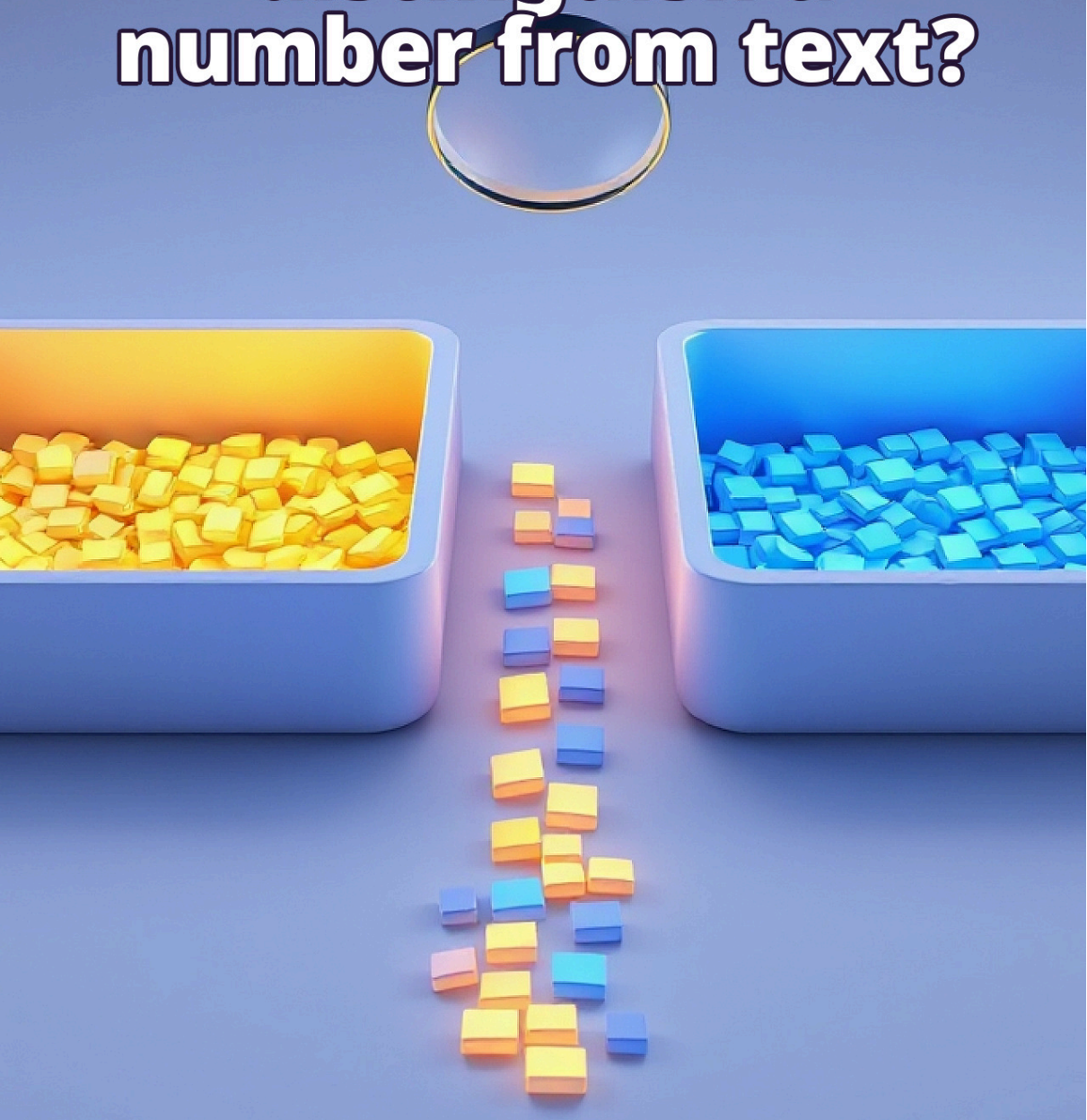
How does a computer distinguish a number from text?

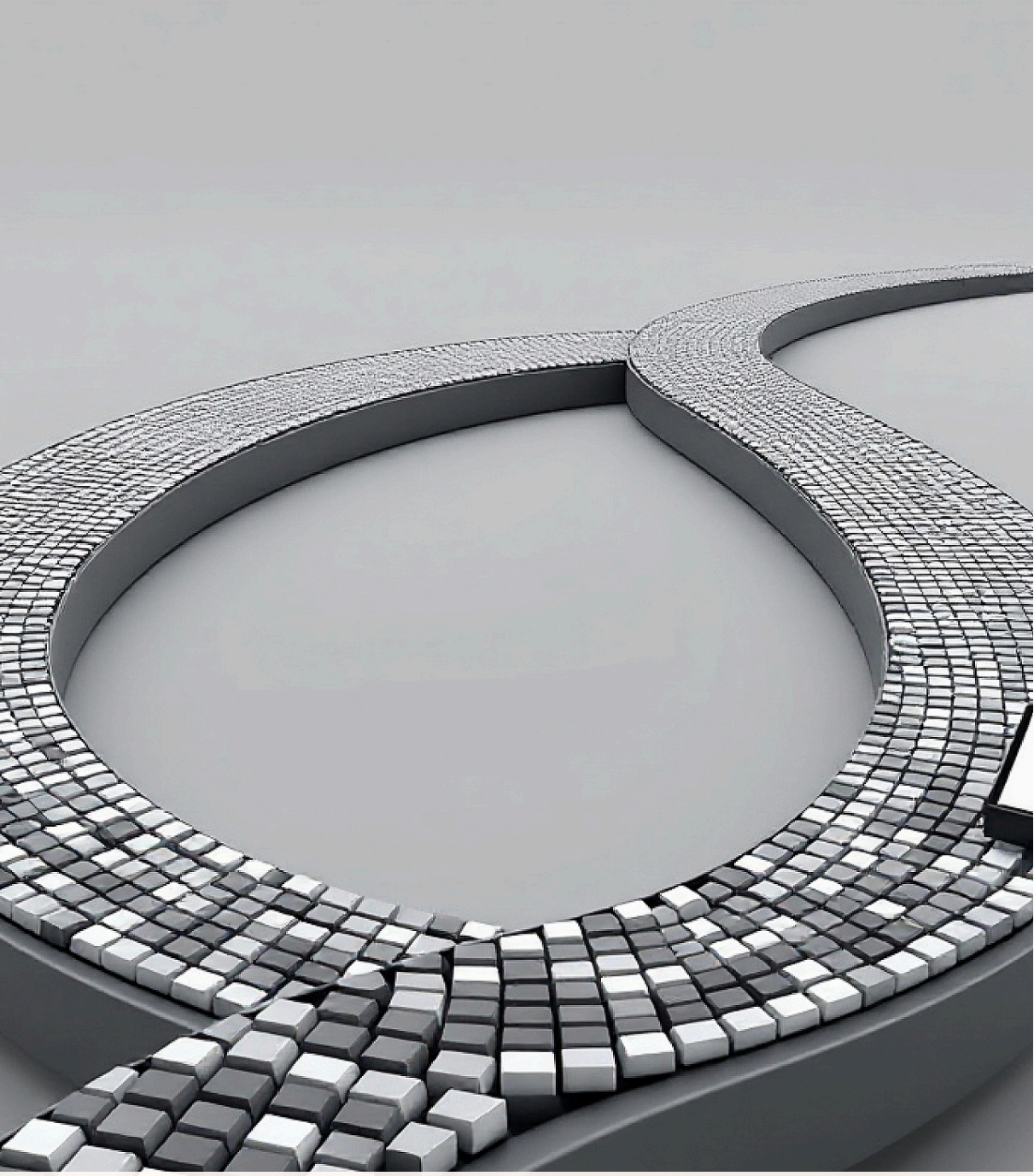


www.yzokul.com.tr

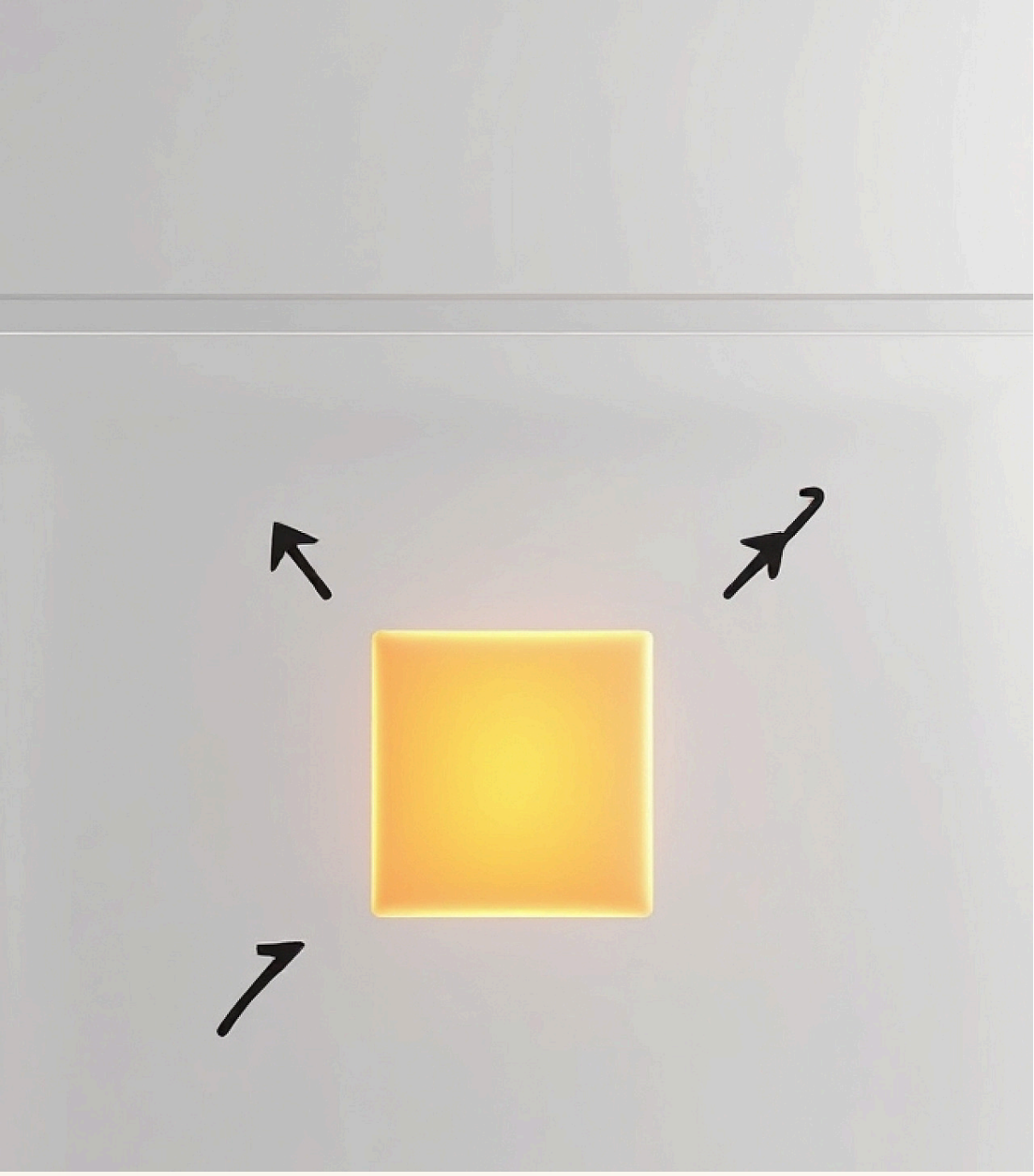
How does a computer distinguish a number from text?



You have one question: how does a computer tell a number from text? The short answer is that the programming language and the runtime system keep track of each value's type by their rules. The language is the set of words and symbols you write; the runtime system is the part that runs your program and carries out those rules. You do not have to check every digit yourself, because the type label does that work.



Inside the machine, everything is stored as bits, which are simply ones and zeros. A number, a letter and a picture all become patterns of these bits. Bits alone do not say what they mean, because the same pattern can stand for different things. Looking only at the bits, you cannot tell whether they mean a number or text. Something else must supply the meaning.



Meaning comes from agreed rules. The same bits can be read as a number or as text depending on which rule you apply. The programming language and the runtime system supply those rules and apply them to every value in your program. When you write a value in a certain way, the rules decide what type it gets. The bits stay the same; the interpretation changes.



Each value in a program carries a type label, such as number or text. This label is the type of the value, and it tells the computer how to treat that value. The label is kept by the language and the runtime system, not by you. When the program needs to act on a value, it first looks at the label to know what kind of thing it is dealing with.



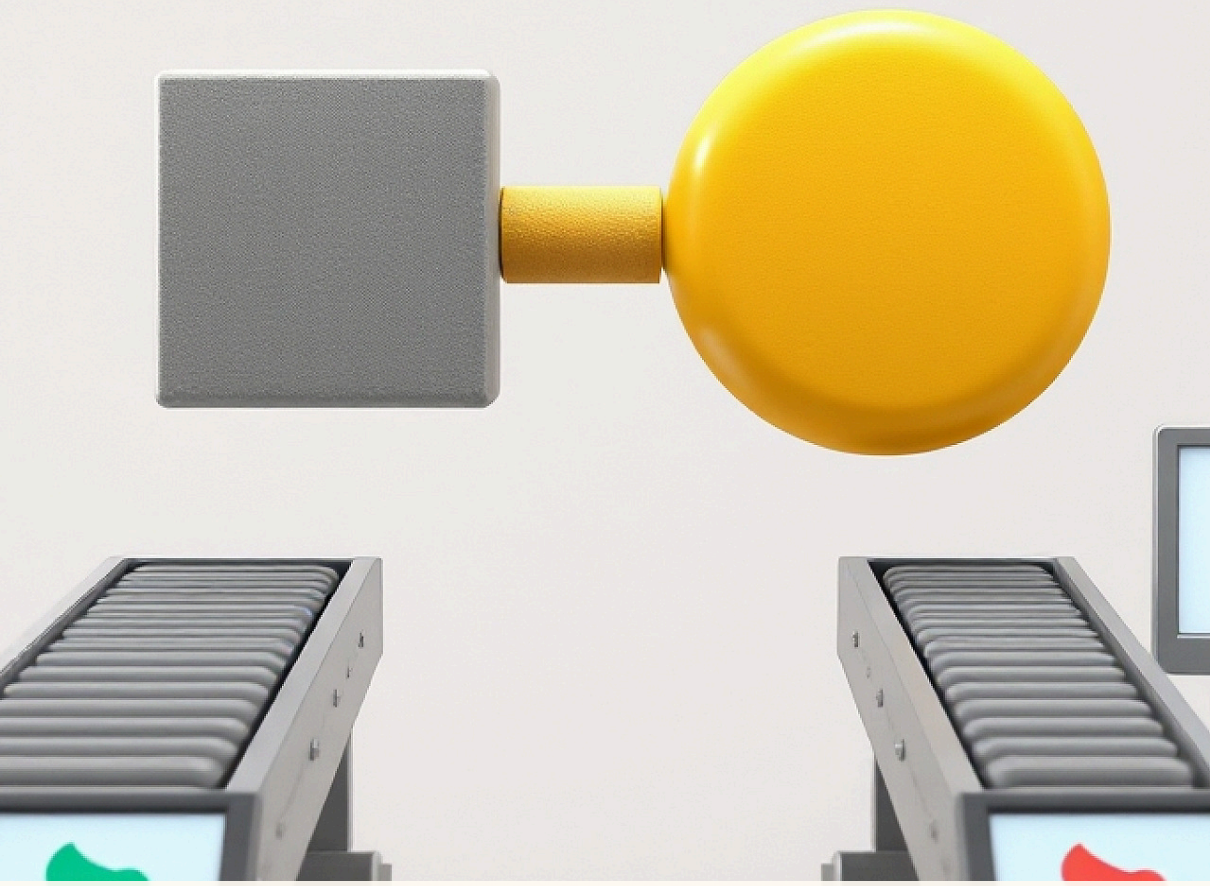
Look at 42 written inside quotation marks, as in "42". In most programming languages, the quotation marks mark it as text, not as a number. The digits are stored, but the type label says text. This means the value is a short piece of text whose characters happen to be the digits four and two. The quotes are the signal that sets this label.



Now look at 42 written without quotation marks. In many programming languages this may be treated as a number, ready for arithmetic. The type label here says number, so the computer expects to add, subtract or compare it. The exact rule depends on the language, but the missing quotes are the usual sign that a number is meant.



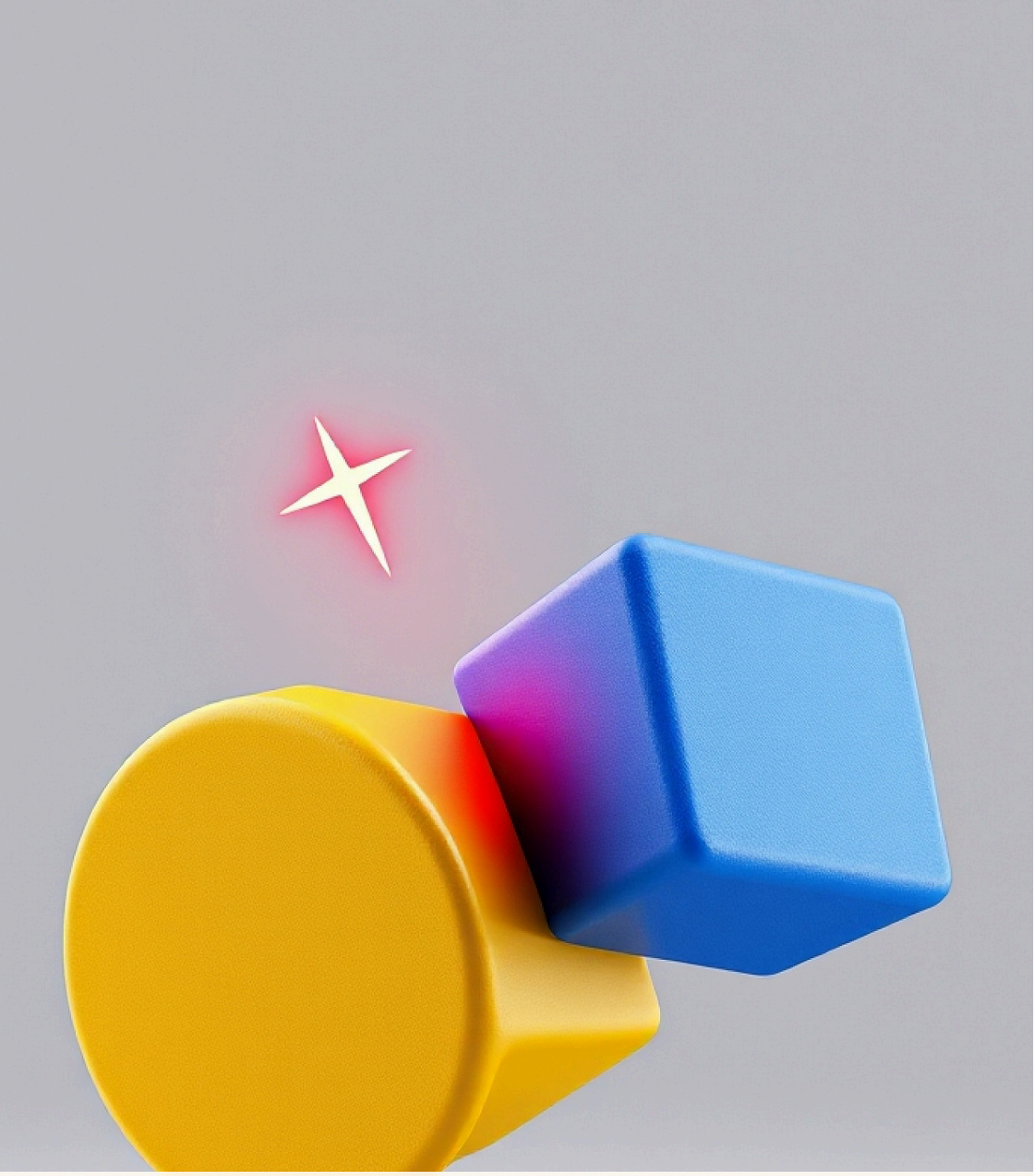
Compare "42" and 42 side by side. The digits look alike, yet the type label differs because of the quotation marks. One value is text and the other may be a number. To your eye they seem the same, but to the language they are different kinds of value. This small difference in writing changes how the computer will treat them.



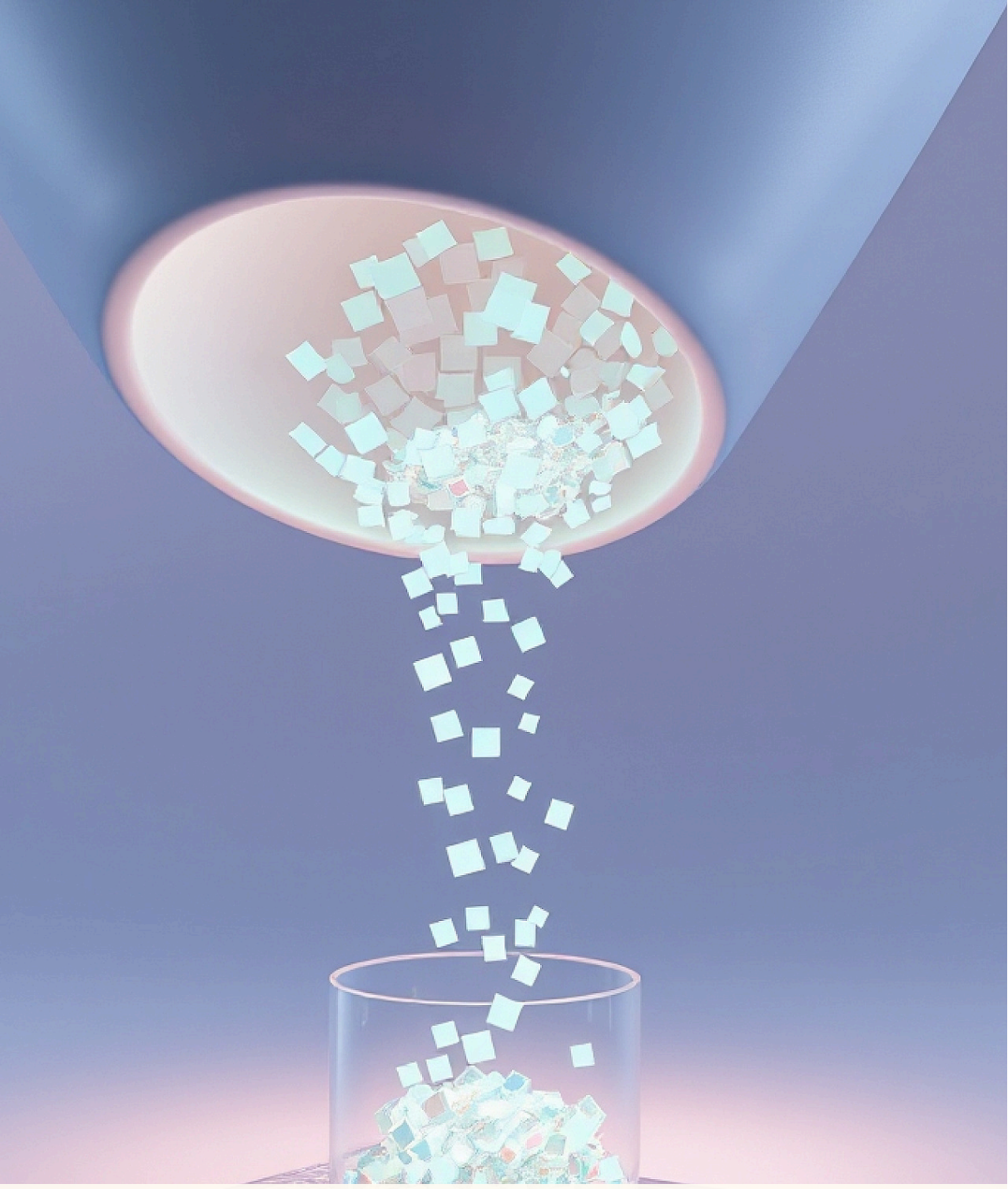
The type decides which operation happens. Two texts can be joined side by side into one longer text, while two numbers are added to give a sum. Joining "ab" and "cd" gives "abcd", but adding 2 and 3 gives 5. The same values under a different label would produce a different result, so the type matters before any operation runs.



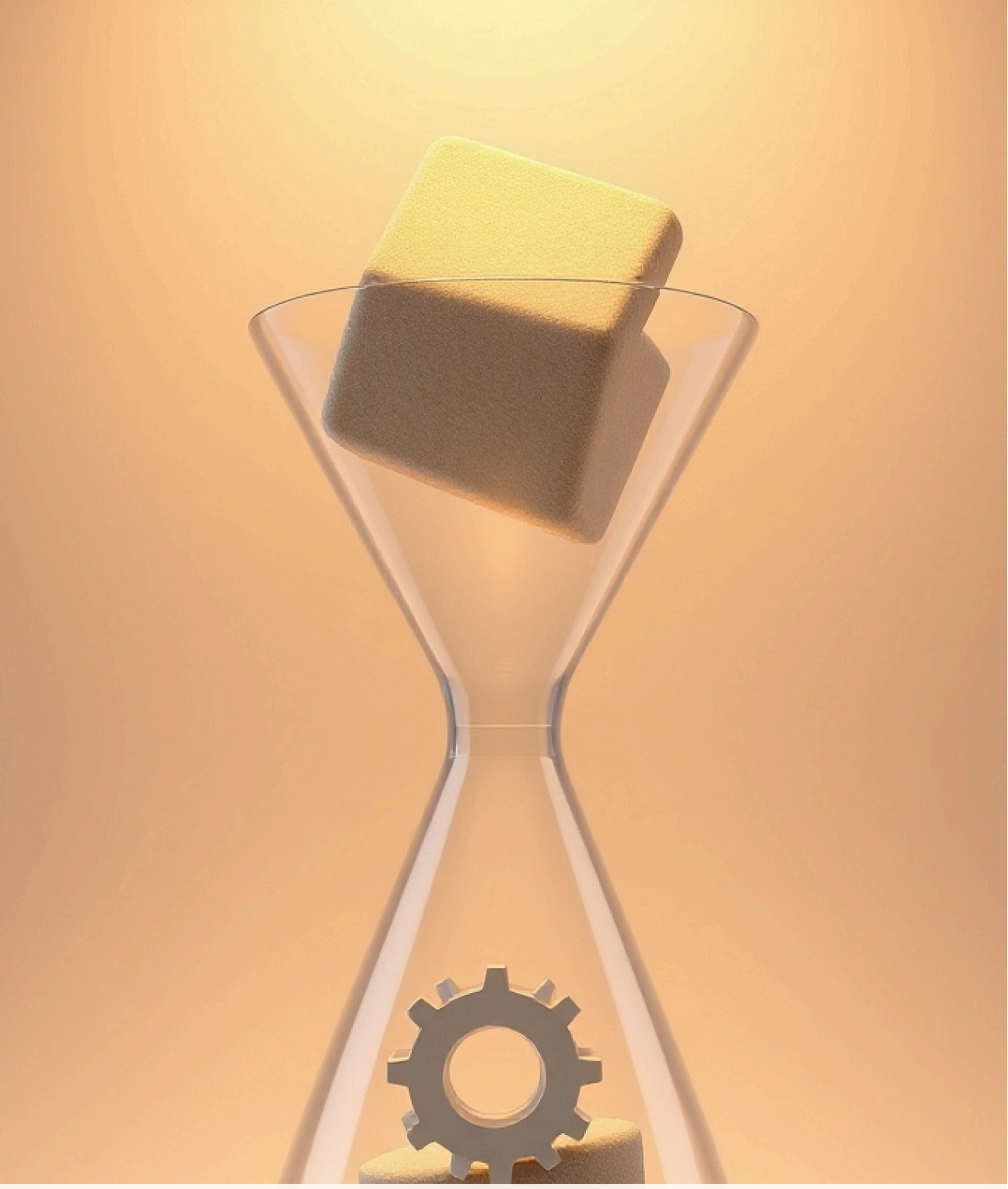
The plus sign does both jobs. It can join text or add numbers, and the computer picks based on the types it is given. If both sides are text, the plus sign joins them. If both sides are numbers, it adds them. You write the same symbol, but the type labels decide which meaning the runtime system applies.



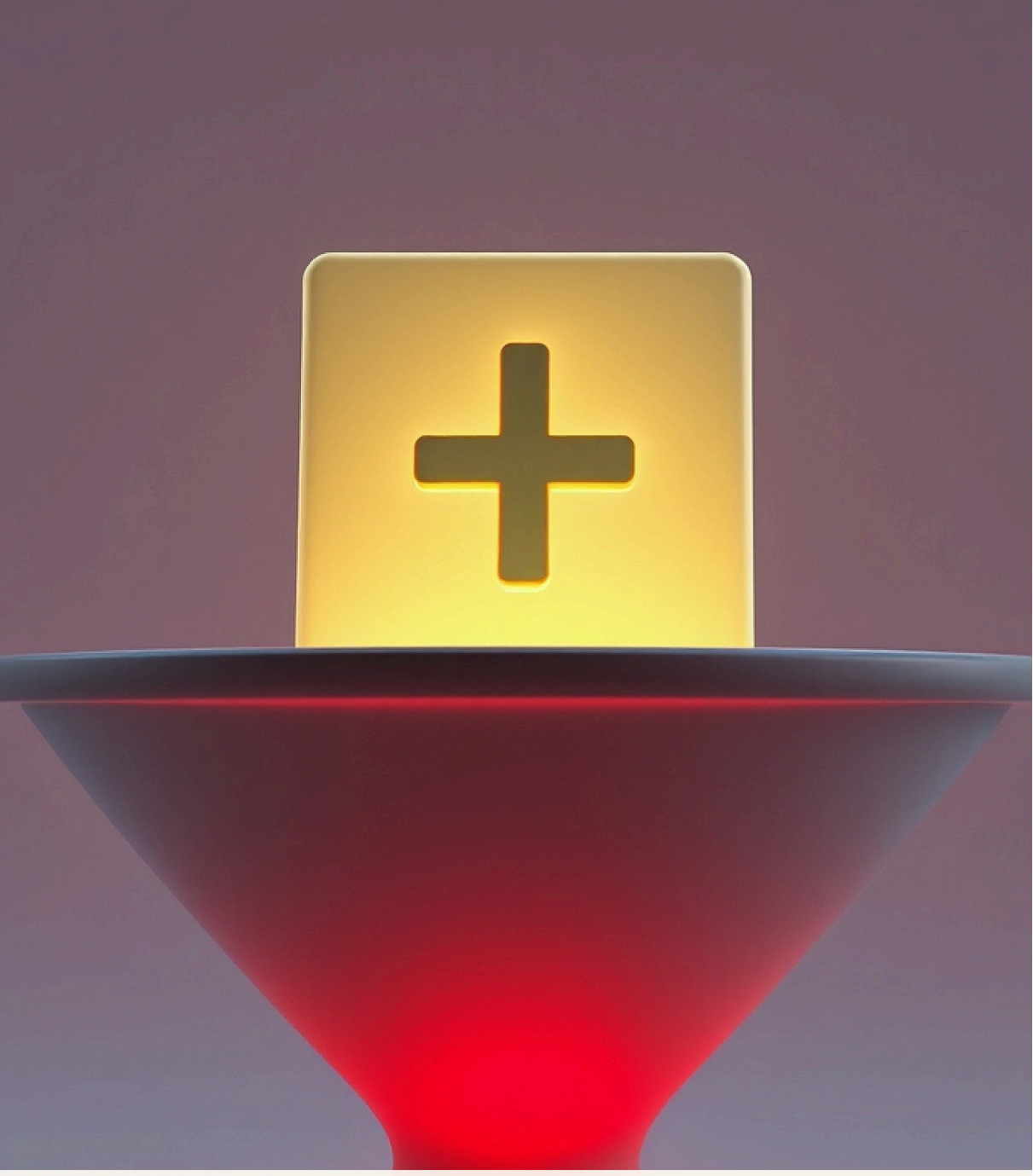
Text plus number is a common mistake. If one value is text and the other is a number, the language must have a rule for what happens, and that rule may surprise you. Some languages refuse and report an error; others quietly convert one side. Because results differ, you should not rely on guessing. Keep both sides the same type.



Input typed by a user usually arrives as text. Even when the user types digits such as 42, the program first receives them as text characters. The keyboard sends characters, not numbers, so the type label on that input says text. Before you calculate with it, you must treat it as text and then convert it.



Text like "42" must be converted to a number before calculating. The language offers a way to do this conversion, often a built-in function such as int or parse. You pass the text in and get a number back. Only after that conversion can you add it to another number and expect a correct sum.



Converting can fail. Turning text such as "hello" into a number does not work, because those characters are not digits. A good program checks for this case instead of assuming success. If the conversion fails, the program can ask the user again or show a clear message. Handling failure keeps the program from producing wrong results.



Watch the quotation marks. A common mistake is leaving quotes around a value meant to be a number, as in "42". The quotes make it text, so adding it to a number may join instead of add, or cause an error. Remove the quotes when you mean a number, and keep them when you mean text.



Check the type before acting. Ask whether a value is text or a number before you choose an operation, especially for anything that came from a user. If it is text and you need arithmetic, convert it first. If it is a number and you need to display it, turn it into text. This avoids surprising results.



Remember the answer: the language and the runtime system track each value's type by their rules. Quotes make text, and no quotes may mean a number. The type label decides whether the plus sign joins or adds. User input often arrives as text, so convert it to a number before you calculate.