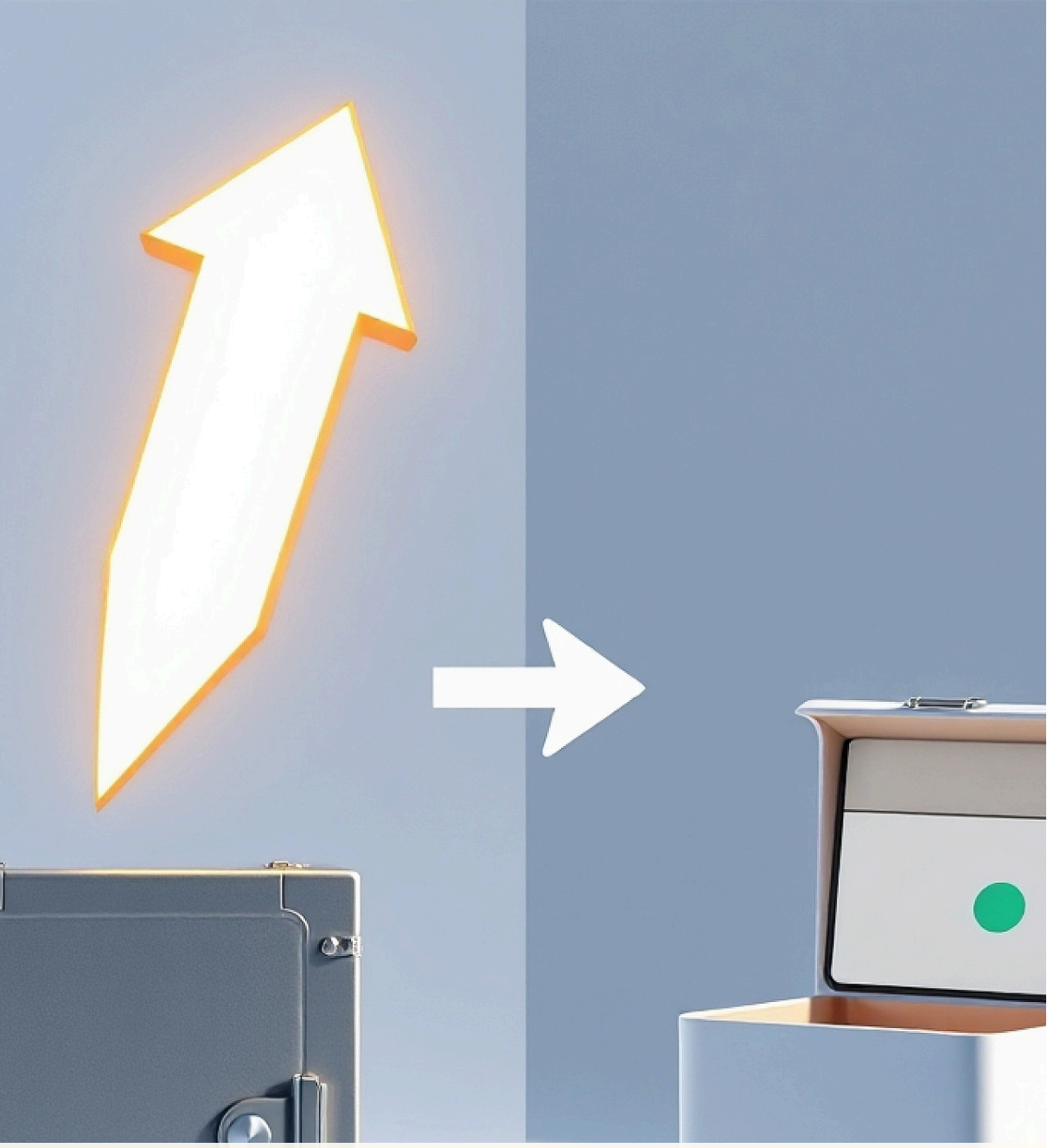
What is the difference between a constant and a variable?



www.yzokul.com.tr

What is the difference between a constant and a variable?



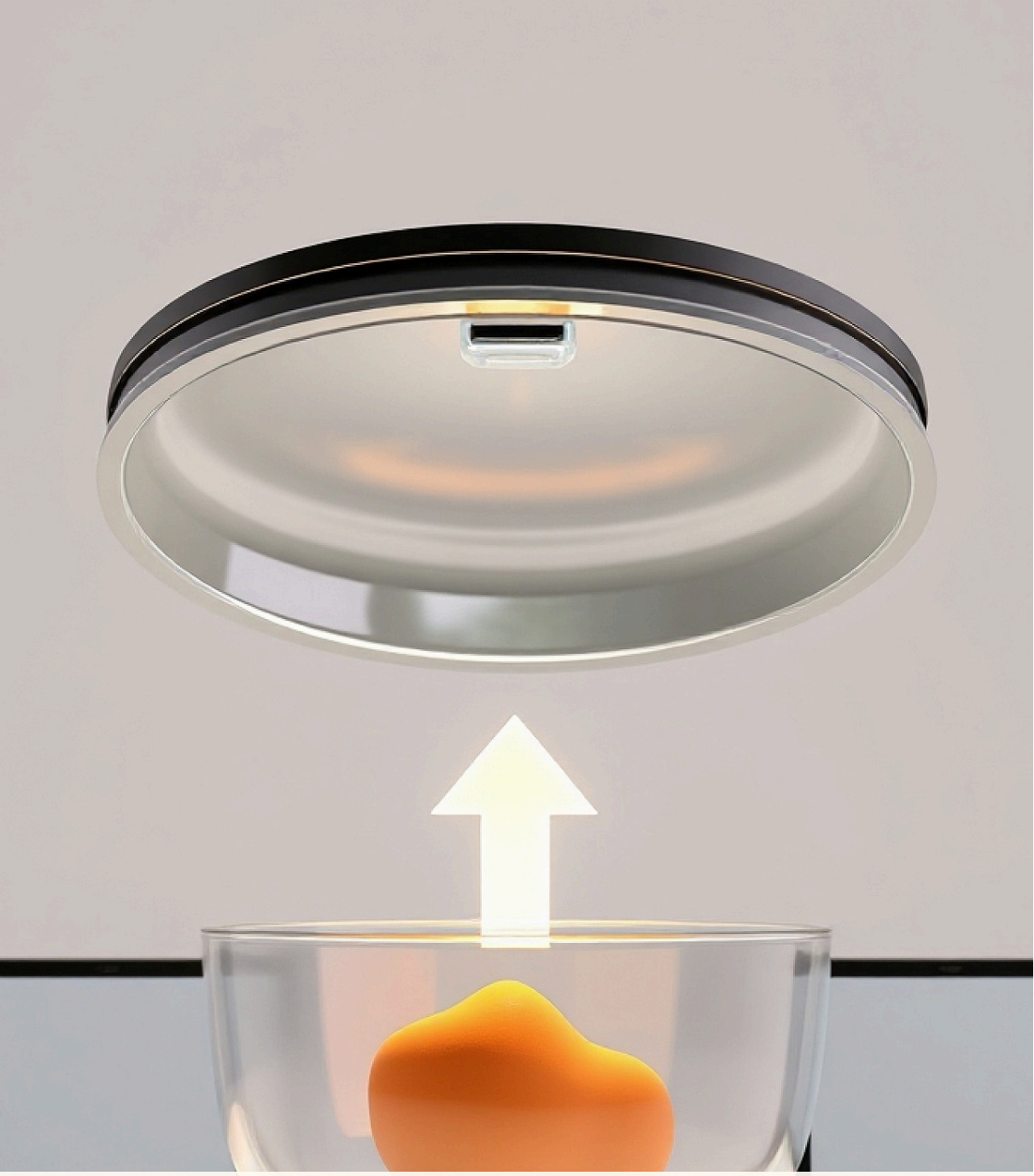
What is the difference between a constant and a variable? The short answer is that a constant cannot be reassigned under the program rules, while a variable can change value. Reassigning means giving a name a new value. This book explains each idea step by step, starting with the basics of values and names. By the end, you will understand the key difference and avoid common mistakes.



A value is simply a piece of information that a program keeps, such as a number, a word, or a true/false flag. Every constant and every variable holds some value. For example, the number 42 is a value, and the word hello is a value. Programs store values so they can use them later. Understanding values is the first step to understanding constants and variables.



Programs give names to values so they can refer to them later. A name is just a label that points to a value. For example, you might name a value score or maximum. When you use the name in your program, it stands for the value it points to. Names make programs easier to read and write. They are not the values themselves; they are references to values.



A variable is a named value that is allowed to change. You can give a variable a new value at any time, and the name will then point to the new value. For example, a variable named score might first point to 0, and later point to 10. The name stays the same, but the value it points to can be replaced. This ability to change is what makes a variable useful.



A constant is a named value that is not allowed to be reassigned. Once you set a constant, you cannot give it a new value under the program rules. For example, a constant named MAX_SCORE might be set to 100 and must stay 100. The name is locked to its value. Trying to reassign a constant will cause an error in many programming languages.



To change a variable, you assign it a new value. The name stays the same, but the value it points to is replaced. For example, if score is 0, you can write score = 10 to make it 10. After this, the name score points to 10 instead of 0. This is called reassignment. Variables are designed to allow this kind of change.



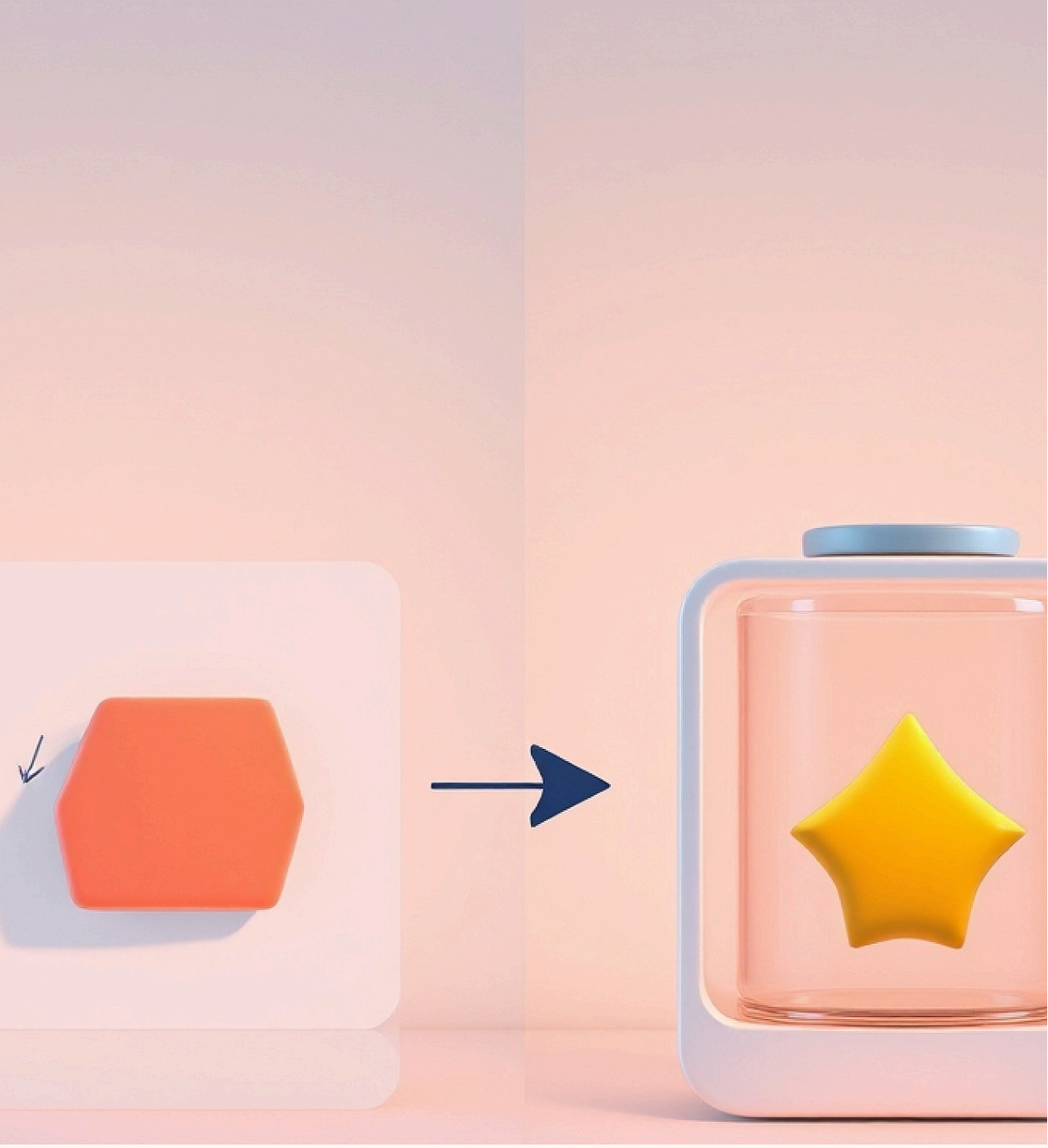
Trying to reassign a constant breaks the program rules. If you define a constant named MAX and set it to 100, you cannot later write MAX = 200. The program will reject this attempt, often with an error message. The picture in your mind should be a locked box: the name is locked to its value. This protection keeps important values from changing accidentally.



Imagine a game where you earn points. Your current score goes up as you play, so it must change. You store the score in a variable, for example score = 0 at the start, then score = score + 10 when you earn points. Because the score changes, a variable is the right choice. The game updates the score many times during play.



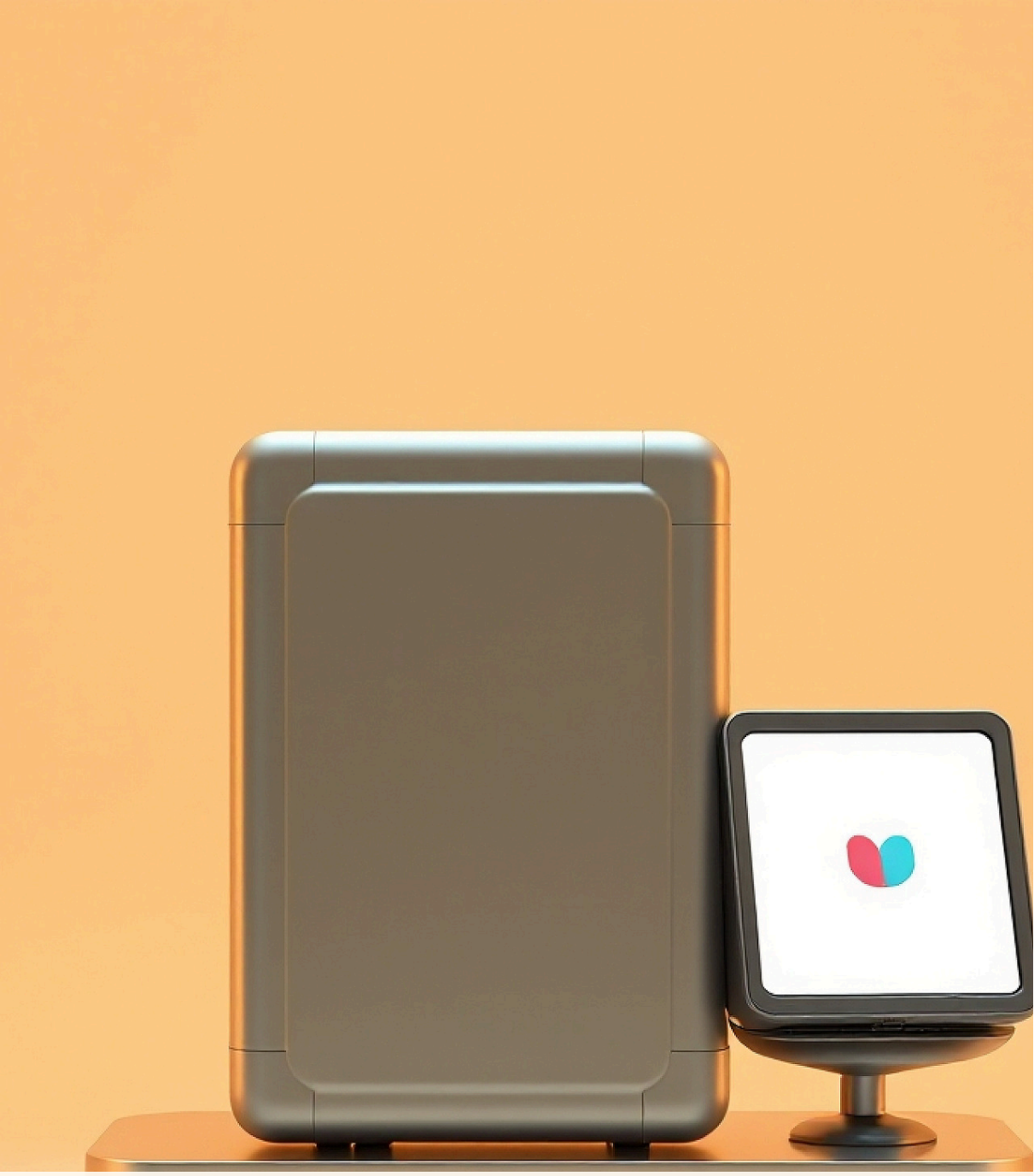
A fixed upper limit, such as the maximum possible score, should not change during the game. You keep it in a separate definition, like MAX_SCORE = 100. This is a constant because it is set once and never reassigned. Using a constant makes it clear that this value is meant to stay fixed. It also prevents accidental changes that could break the game.



Reassigning means pointing a name at a new value. Changing contents means altering what an existing object holds. These are different actions. For example, reassigning a variable score from 0 to 10 points the name to a new number. Changing contents would be like modifying a list by adding an item, without pointing the name to a new list. Keep these two ideas separate.



In some languages, a constant name may be bound to an object whose contents can still change. The name cannot be reassigned to a different object, but the object itself can be modified. For example, a constant list can have items added or removed, but you cannot make the constant name point to a new list. This is a common source of confusion, so remember: the constant name is fixed, but the object's contents may not be.



Constants protect values that should stay fixed, while variables allow values that must change. This helps programs stay clear and correct. For example, using a constant for a tax rate ensures it is not changed by accident. Using a variable for a user's age allows it to be updated. Choosing the right one makes your code easier to understand and less error-prone.



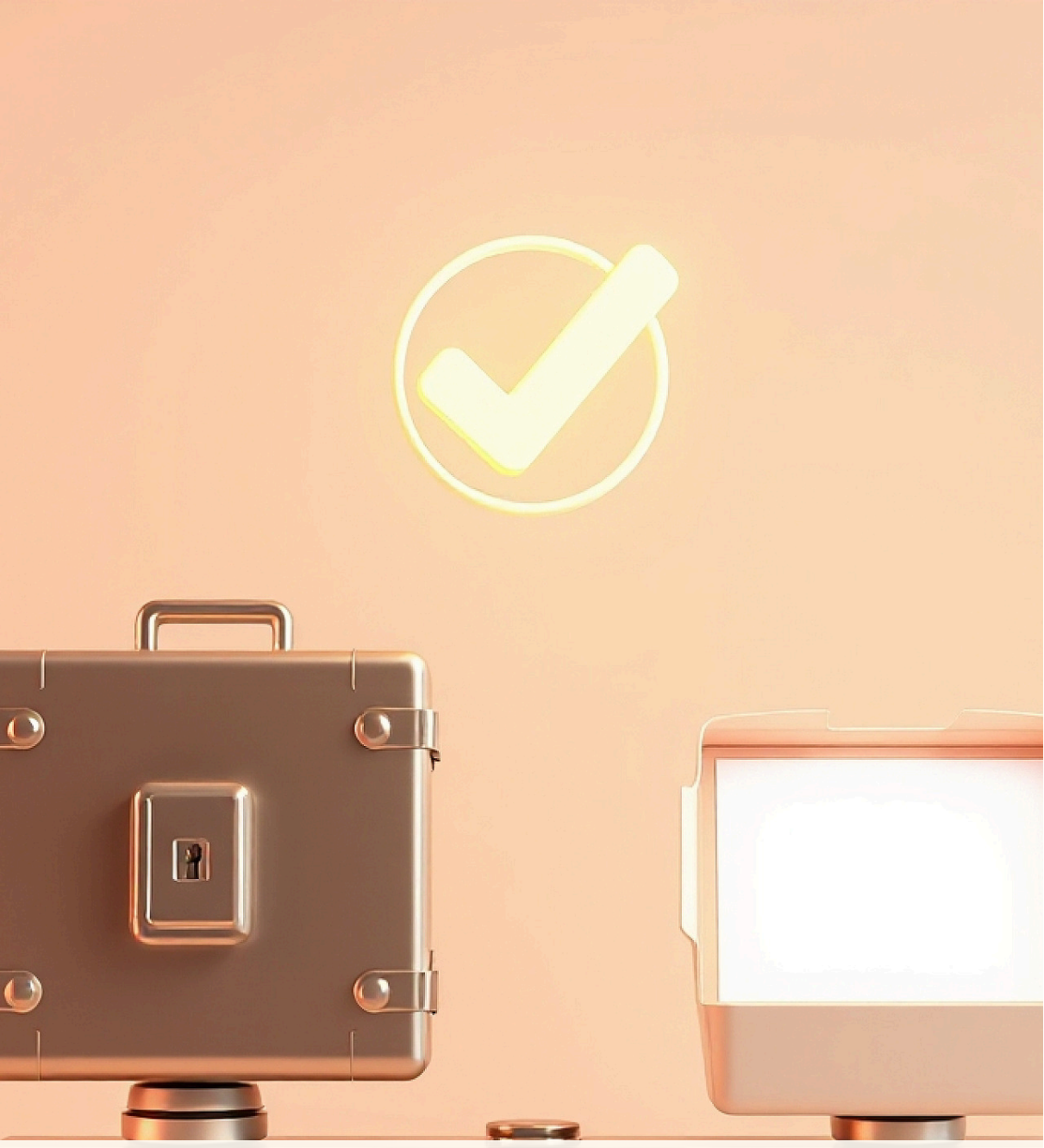
A common mistake is thinking a constant's contents can never change. In reality, reassignment and changing contents are not the same thing. A constant cannot be reassigned, but if it points to a mutable object, that object's contents can often be changed. For example, a constant list can have items added. So, do not assume a constant means everything about it is frozen.



Another mistake is thinking a variable can always be changed in every situation. Program rules still apply to when and how a variable may be reassigned. For example, a variable might be read-only in certain contexts, or its type might restrict what values it can hold. Also, some languages have different rules for variables. Always check the rules of your programming language.



Confusing a name with the value it points to is a common error. The name is just a label; the value is what the program actually stores. For example, the name score is not the number 10; it points to 10. If you change the value, the name still points to the new value. Keep in mind that names are references, not the values themselves.



To sum up: a constant cannot be reassigned, a variable can change value, and reassigning differs from changing an object's contents. Remember that a constant name may point to an object whose contents can change. Use constants for fixed values and variables for values that need to change. These key points will help you use constants and variables correctly.