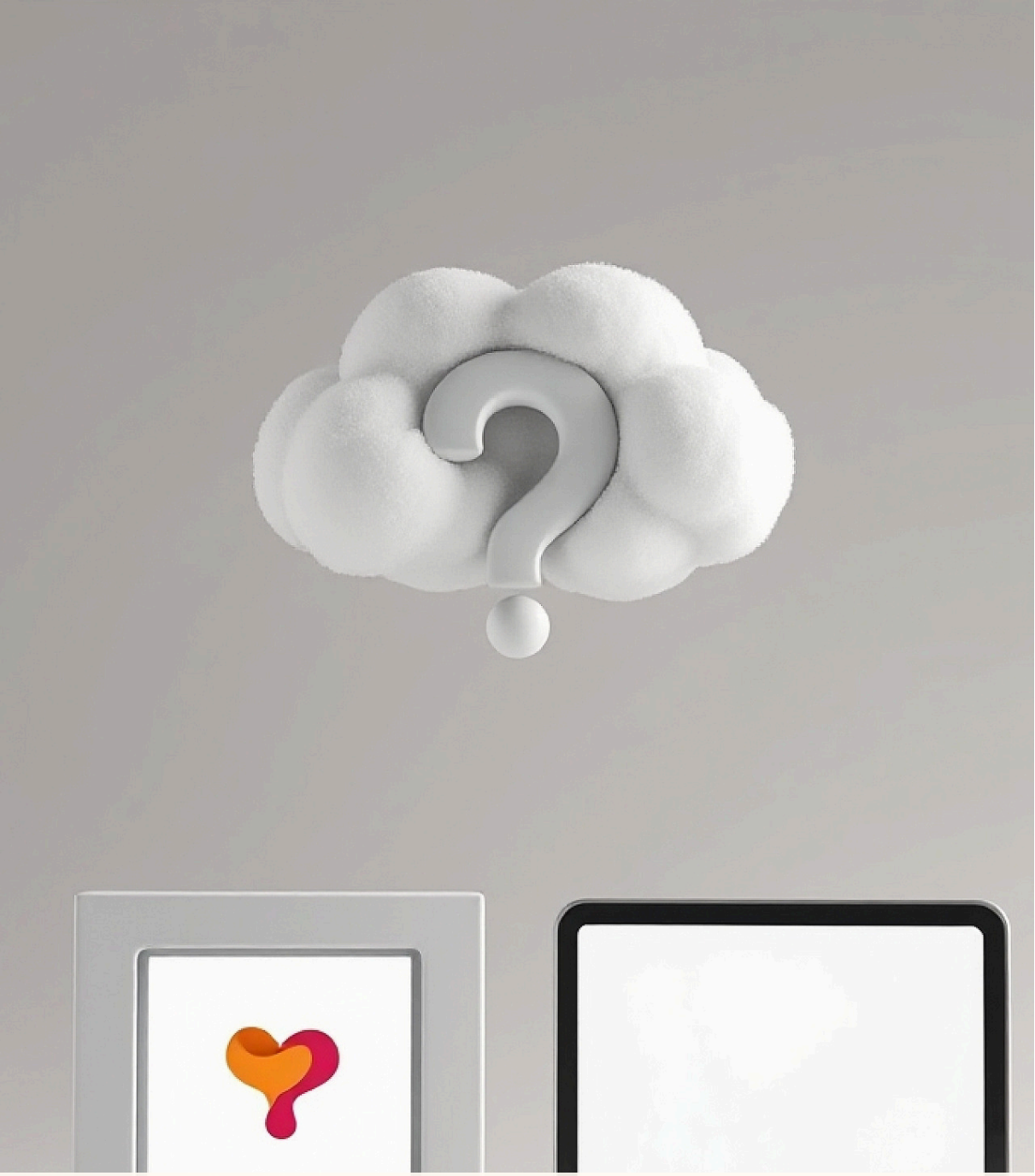
What is the difference between a local and a global variable?



www.yzokul.com.tr

What is the difference between a local and a global variable?



You read the question: what is the difference between a local and a global variable? The short answer is that a local variable works only in the small section where it is defined, while a global variable can be reached from a broader area of the program. This answer book explains each idea step by step with simple examples.



A variable is a named box in the computer's memory that holds a value, like a number. You can put a value in the box, read it later, and change it whenever you need to. For example, a box named score might hold the number 10. The name lets you refer to that value without remembering where it is stored.



A program is not one flat list of instructions. It has sections, such as functions that each do one job. A function is a small named block of instructions that can be used again and again. For example, a function called add might take two numbers and return their sum. Sections help you organize your code.



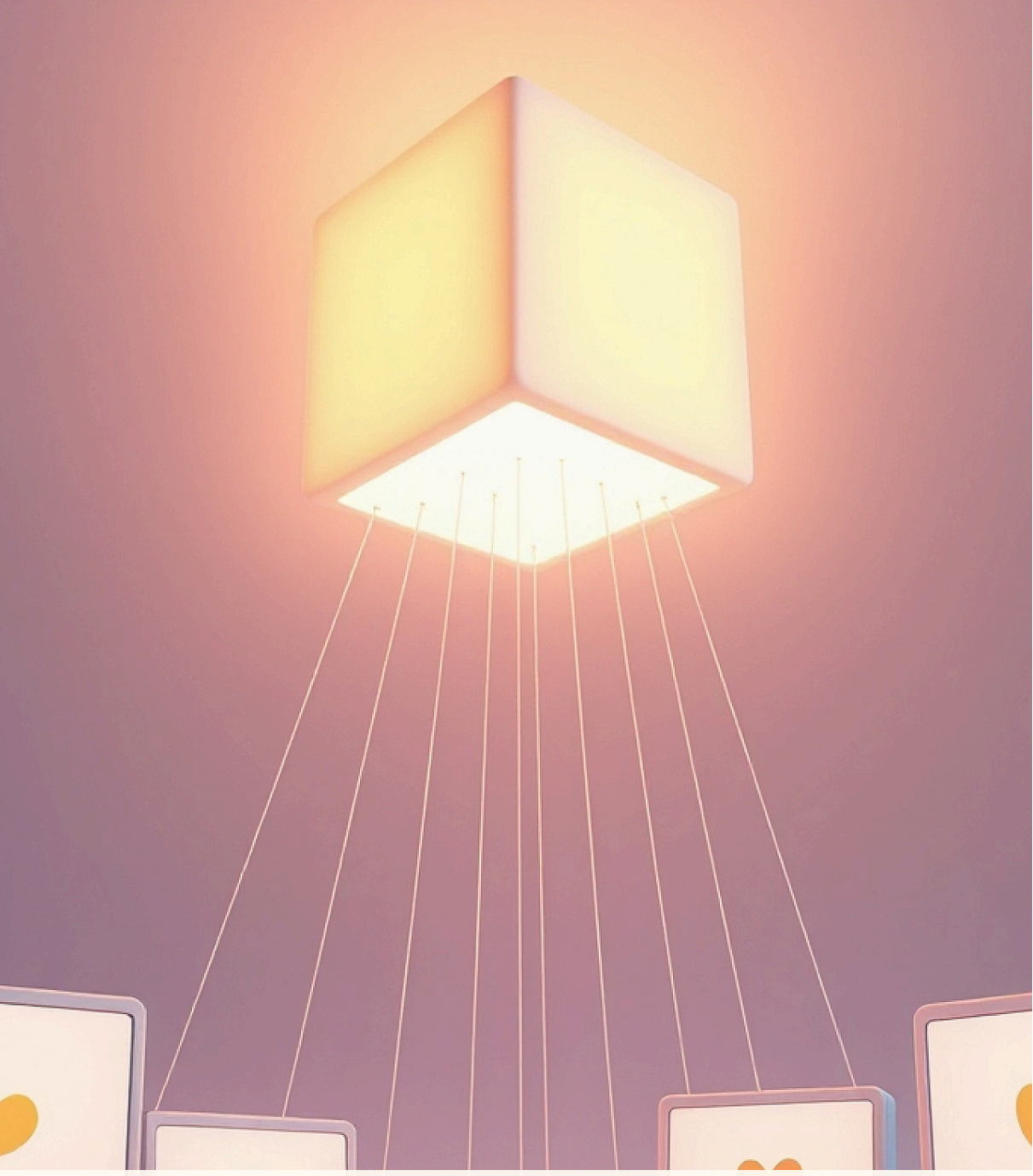
Every variable is created somewhere in the program text, inside some section. That place where it is defined decides where it can be used. If you define a variable inside a function, it belongs to that function. If you define it outside all functions, it belongs to the whole program. The place of definition sets the rules.



A local variable is defined inside a function, for example a temporary calculation number. It exists only while that function runs and only inside that function. For example, inside a function that adds two numbers, you might create a local variable named total to hold the result. When the function finishes, total disappears.



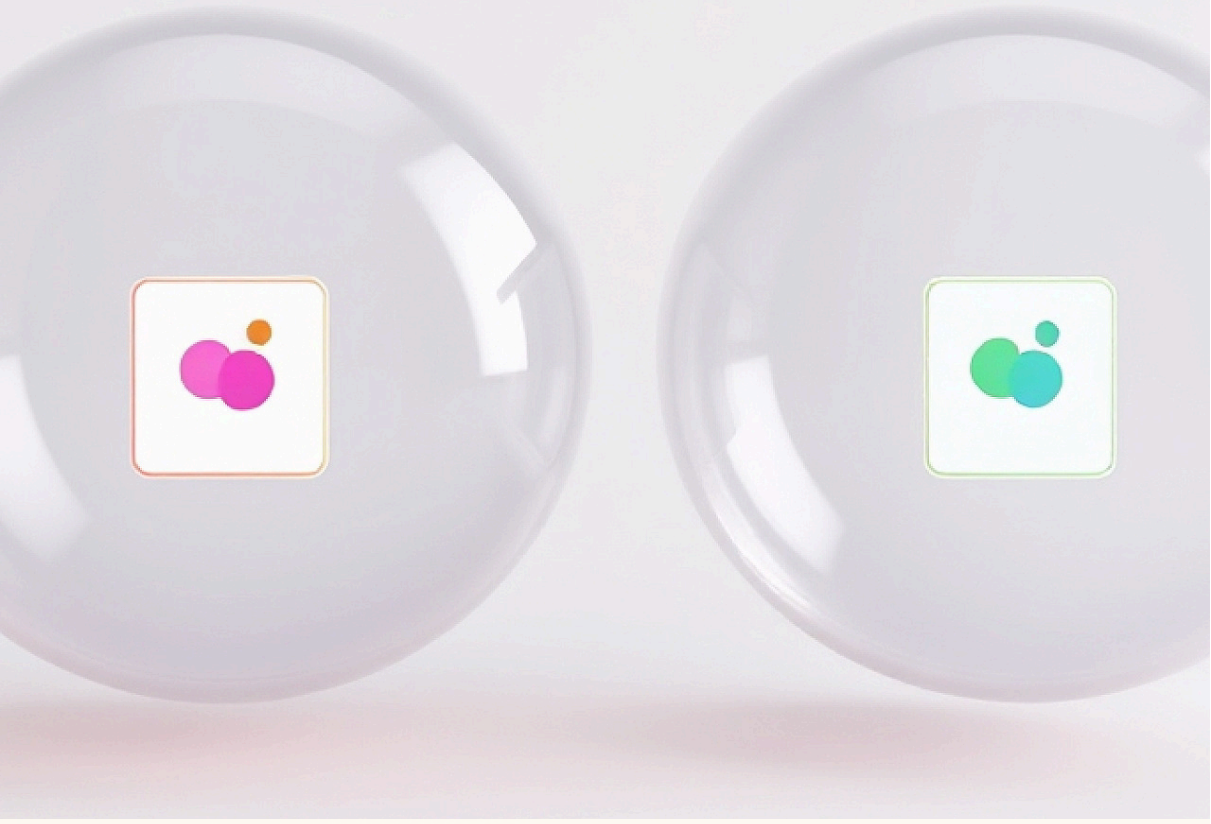
Code outside the function cannot see or use a local variable. If you try to use it from another section, the program treats it as unknown, because it belongs only to that small section. For example, if you define total inside a function, you cannot print total from outside that function. The name simply does not exist there.



A global variable is defined outside any function, at the top level of the program. It is created once and stays available while the program runs. For example, you might define a global variable named app_name at the top of your file. Every function in the program can then read that value. It lives for the whole program.



Many different sections of the program can read and change a global variable. That wider reach is exactly what makes it global. For example, one function might set app_name to a new value, and another function can read that new value later. The change is visible everywhere, because the variable is shared.



Two functions can each have their own local variable with the same name, and they do not clash. Each local box belongs to its own function only. For example, function one might have a local variable named count, and function two might also have a local variable named count. They are separate boxes that never interfere.



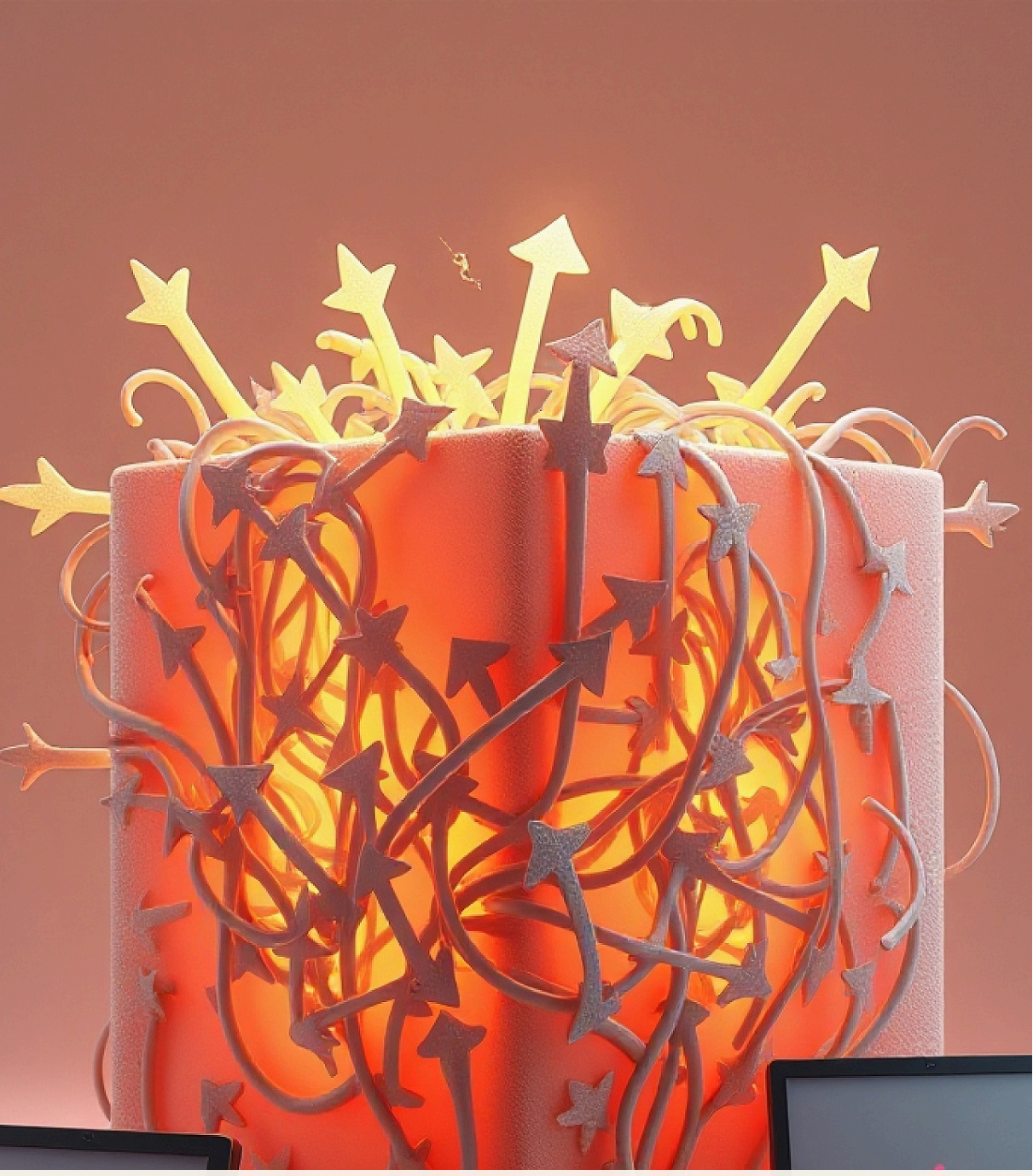
The access area of a variable is called its scope. A local variable has a small scope; a global variable has a broad scope. For example, a variable defined inside a function has scope limited to that function. A variable defined at the top level has scope covering the whole program. Scope tells you where a name can be used.



The programming language's rules decide exactly where a variable can be used. You do not choose the scope freely; the place of definition sets it. For example, in most languages, defining a variable inside a function makes it local automatically. You do not need to declare it as local. The language enforces these rules for you.



A function's temporary calculation number is a good fit for a local variable. It is used briefly and then forgotten, so nothing else needs it. For example, a function that calculates a discount might use a local variable named temp to hold an intermediate result. After the function returns, temp is no longer needed. Keeping it local keeps the program clean.



Making all information global lets different sections affect each other unexpectedly. One part changes the shared value and another part breaks without an obvious reason. For example, if two functions both change a global variable named counter, the final value depends on the order they run. This can cause bugs that are hard to find.



A beginner might try to use a function's local variable from outside the function and get an error. The fix is to define the variable where it is actually needed, or pass the value in and out. For example, if you need a result outside a function, return it from the function instead of trying to read the local variable directly.



A beginner might make every variable global to avoid errors, and then watch parts interfere with each other. The habit to build is: keep things local unless something truly must be shared. For example, if only one function uses a value, define it inside that function. Make it global only when several sections really need the same value.



A local variable works only in the section where it is defined, a global variable can be reached from a broader area, and scope is the name for that access area. Remember to keep variables local by default and make them global only when sharing is really needed. This simple habit prevents many confusing bugs.