What is a runtime error?



www.yzokul.com.tr

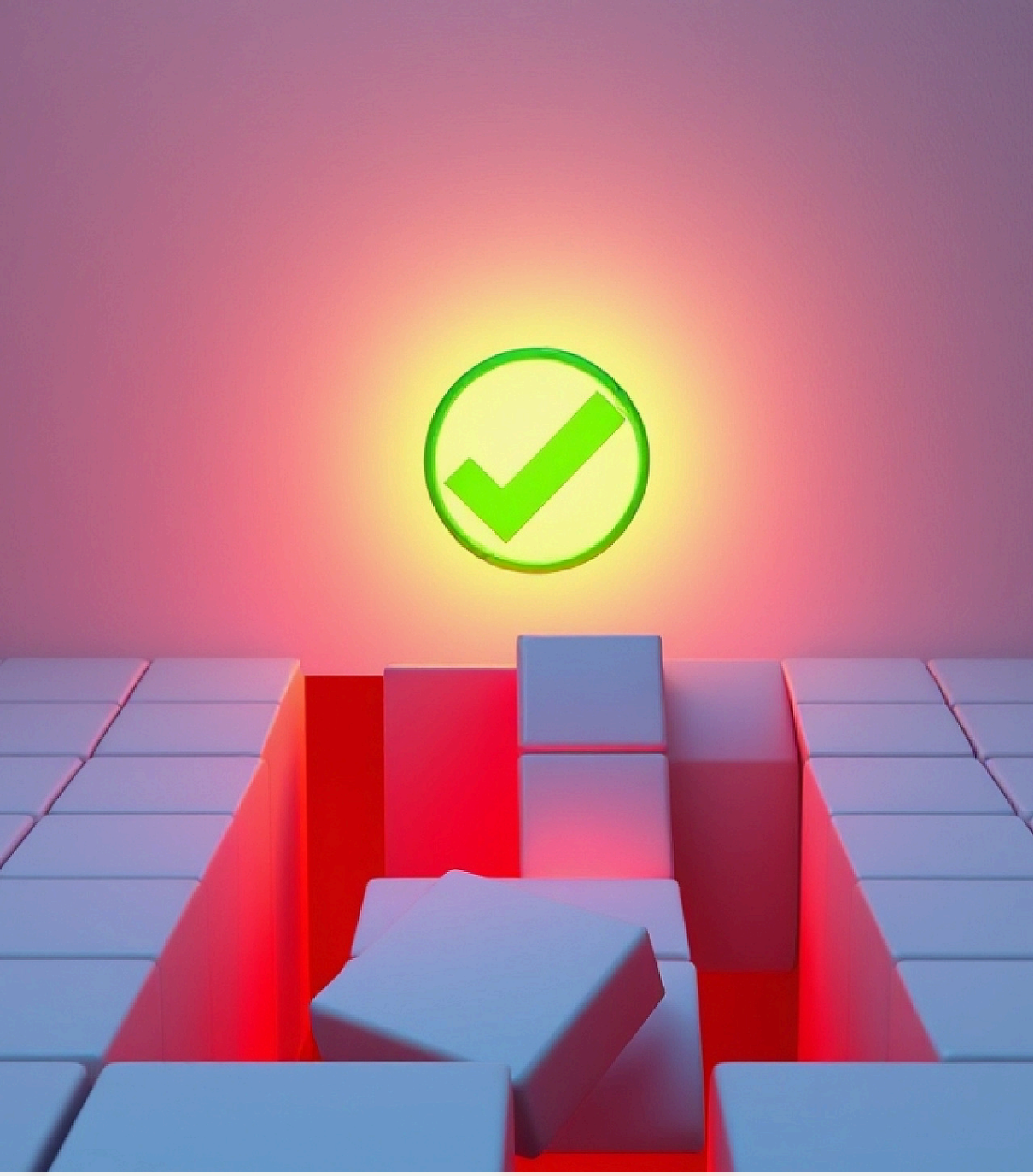
What is a runtime error?



You ask: what is a runtime error? The short answer is that it is a problem that happens while the program is actually running, not while it is being written. This means the error appears during execution, when the computer follows the instructions. It is different from a mistake in the written text.
Understanding this distinction is the first step to handling such errors.



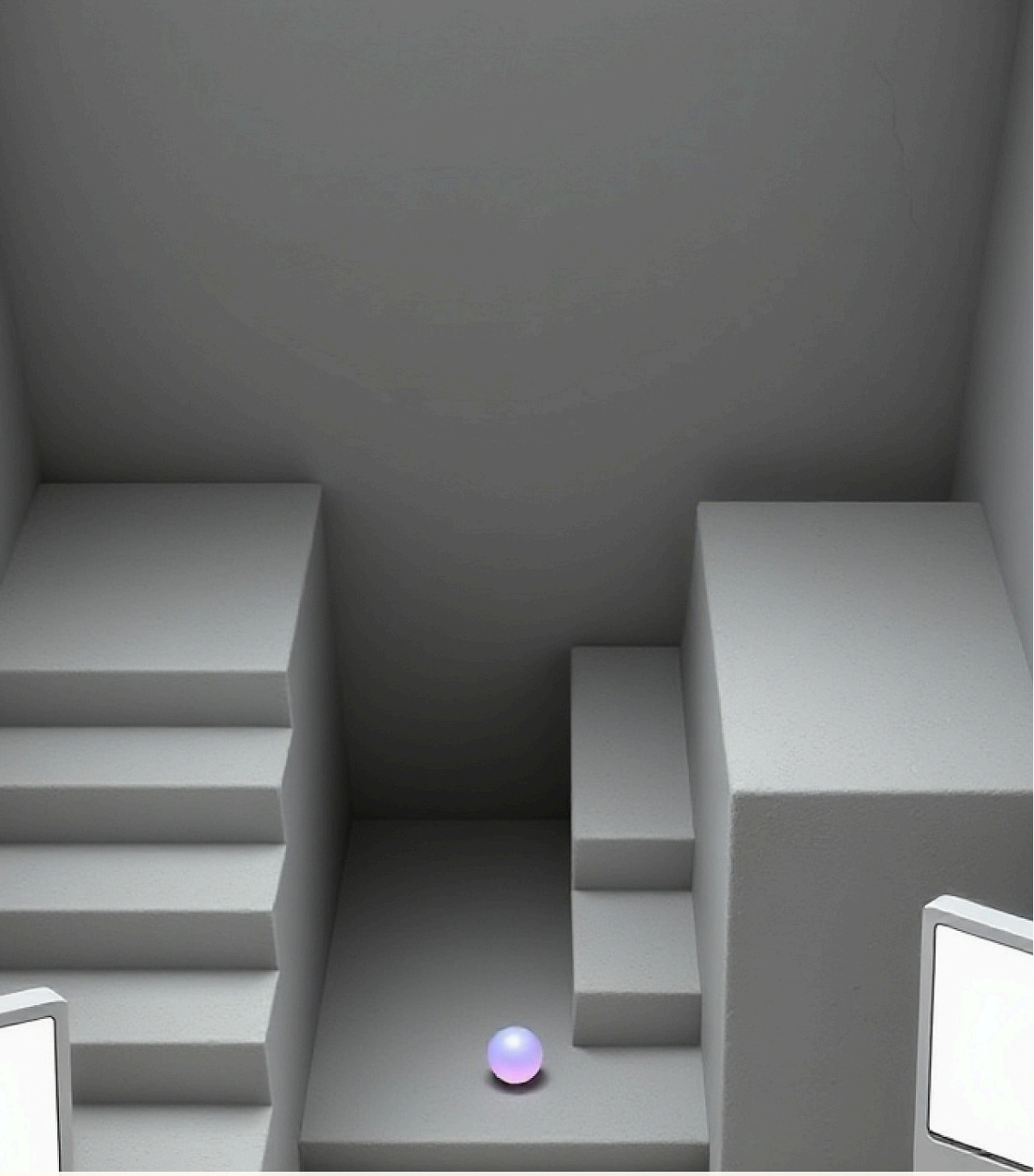
A program is first written as text, then run by the computer. These are two separate phases: writing and running. An error can belong to either phase. Writing errors happen when the text breaks the rules of the programming language. Runtime errors happen later, during the running phase, when the computer executes the instructions. You need to know which phase an error comes from.



Code can follow all the writing rules and still fail later. The writing rules only check the text, not what happens when it runs. For example, a program might be written perfectly according to the language, but when it runs, it tries to open a file that is missing. The text is correct, but the situation at run time is invalid. This is a runtime error.



Picture the computer following the instructions one by one while the program runs. A runtime error appears during this following of instructions. It interrupts the normal flow because something invalid happens at that moment. For instance, the program might try to divide a number by zero, which is not allowed. The computer cannot continue normally, so it stops with an error.



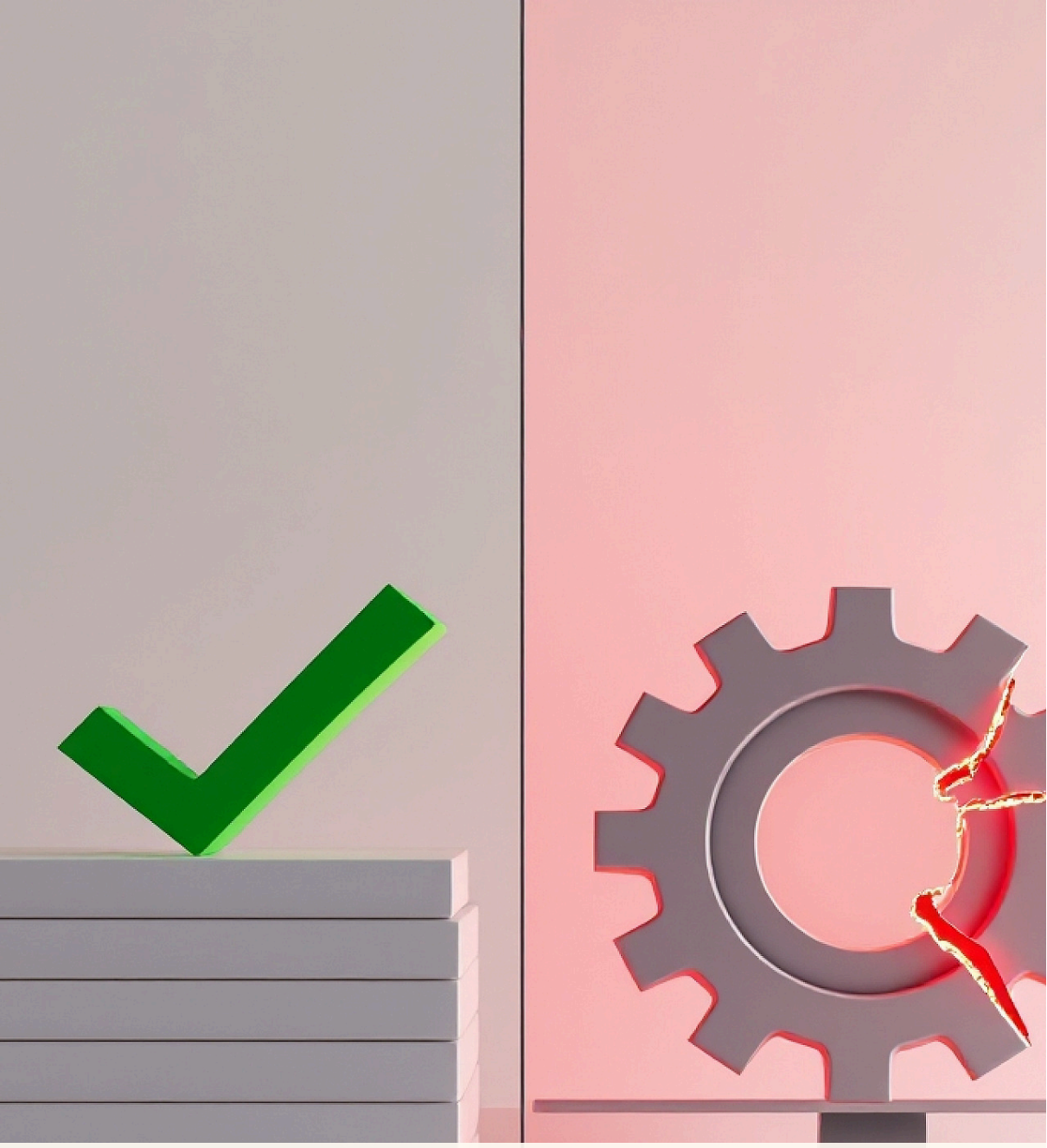
Think of a program as a list of steps, and one step going wrong in the middle. The error stops or interrupts the normal flow at that moment. For example, step three might be to read data from a file, but the file is not there. The program cannot complete that step, so it stops and reports a runtime error. The steps before and after are not reached.



Consider a program that tries to open a file that does not exist. The code is written correctly, but the situation at run time is invalid. The file might have been deleted or never created. When the program runs, it asks the operating system for the file, and the system says it is not found. This causes a runtime error because the expected file is missing.



Consider a program that tries to access a location it is not allowed to use. The text is fine, but the real situation during running causes an error. For example, the program might try to read a system file that requires special permissions. The operating system denies access, and the program receives an error. This is a runtime error because the code itself is correct.



Compare a writing mistake with a runtime error to see the difference. A writing mistake breaks the rules of the language, like a misspelled keyword or a missing parenthesis. The computer catches it before running. A runtime error breaks the real situation, like a missing file, even though the text is correct. They belong to different phases and need different fixes.



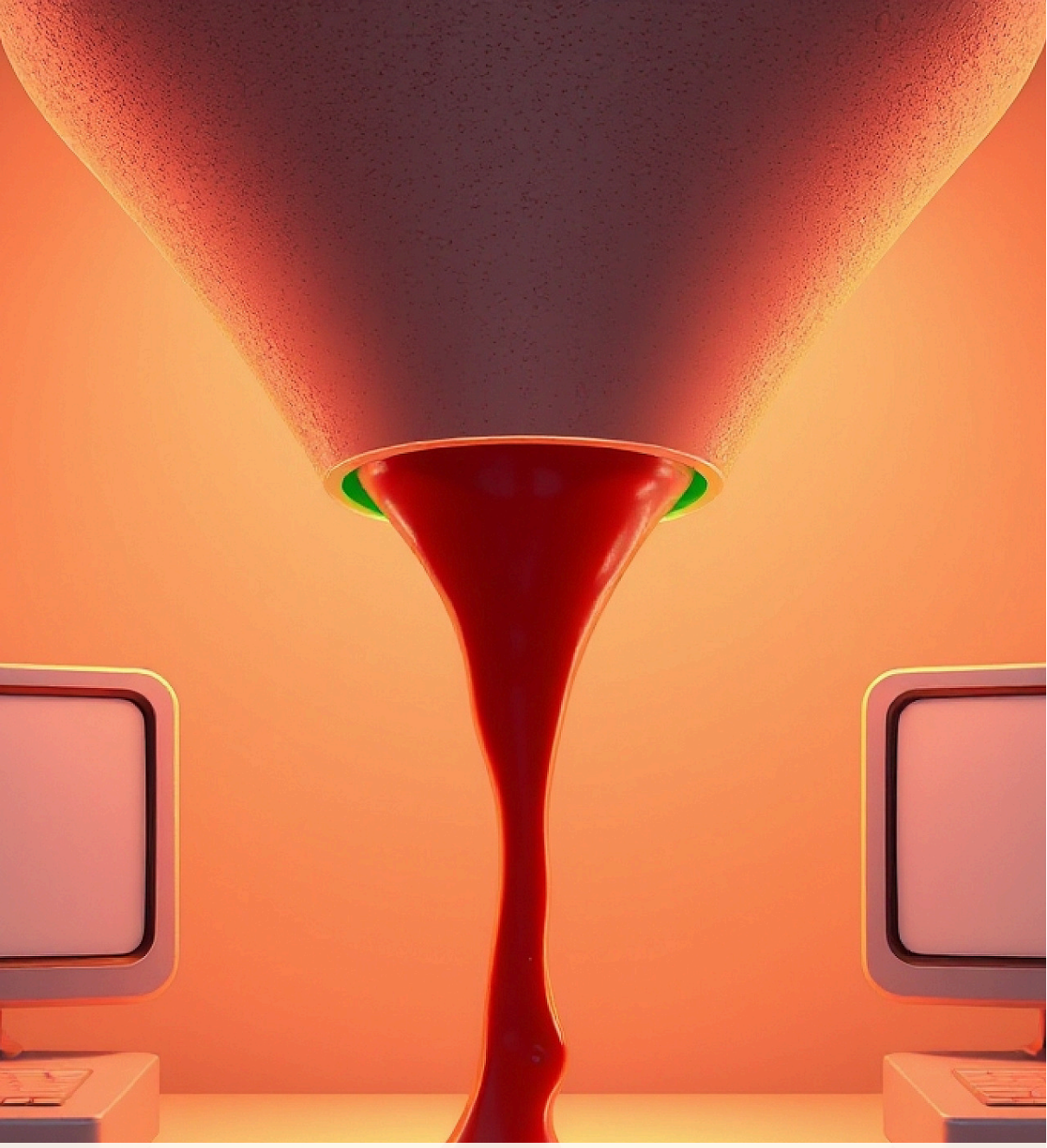
The real world can be different from what the program expects. Files can be missing, places can be forbidden, and data can be in an unexpected format. These invalid situations will occur, no matter how carefully you write the code. Accepting this fact is important because it means you must plan for problems during running, not just hope they never happen.



Such situations must be considered in advance, while writing the program. The programmer plans for what could go wrong during running. This means thinking about possible failures, like a file not existing or a network being down. By anticipating these issues, you can write code that handles them gracefully instead of crashing. Planning ahead is a key skill.



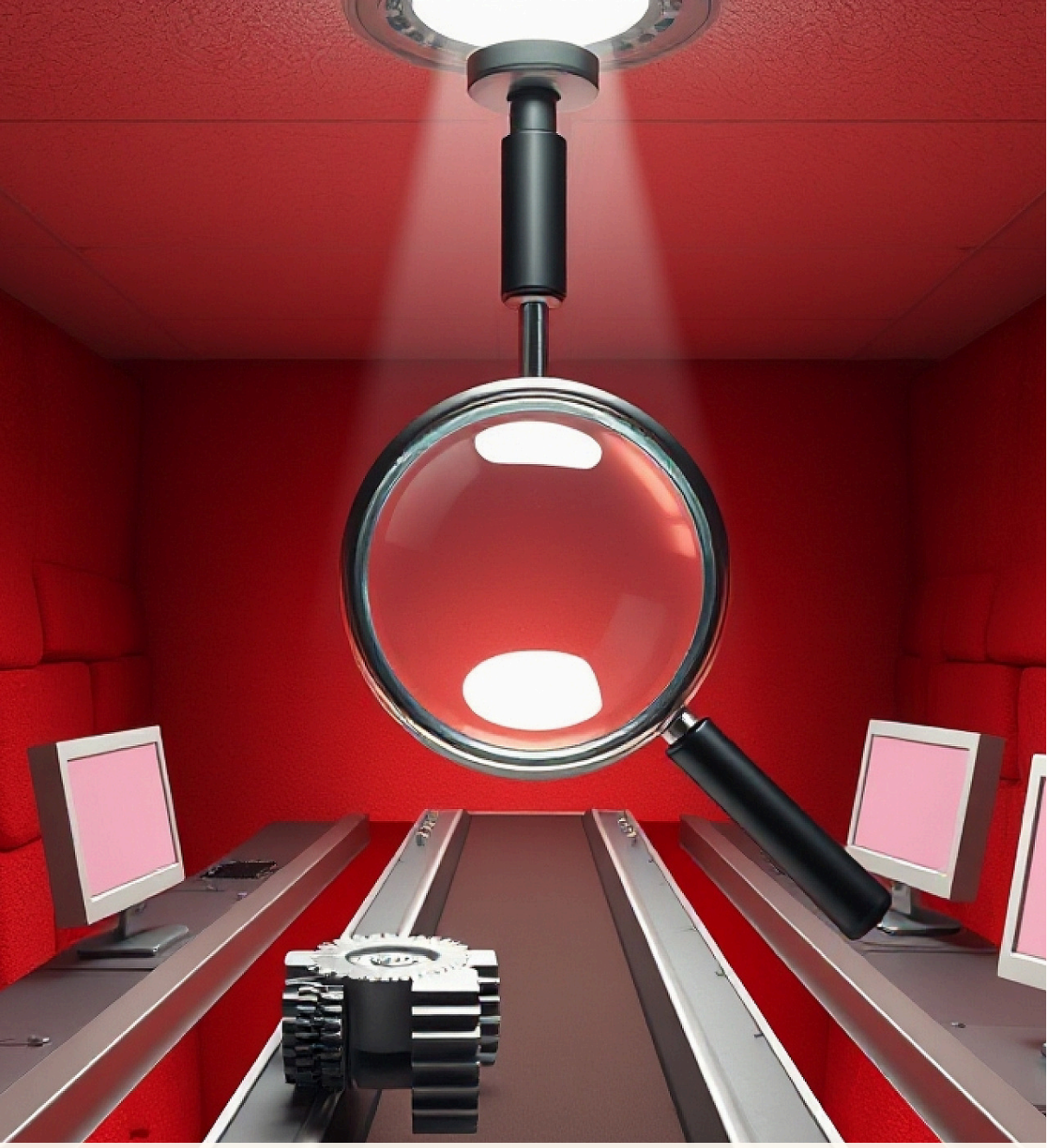
Place a check before a risky step, such as testing whether a file exists first. The check prevents the invalid situation from turning into a crash. For example, before opening a file, the program can ask the operating system if the file is there. If it is not, the program can skip the step or show a message. This simple check avoids a runtime error.



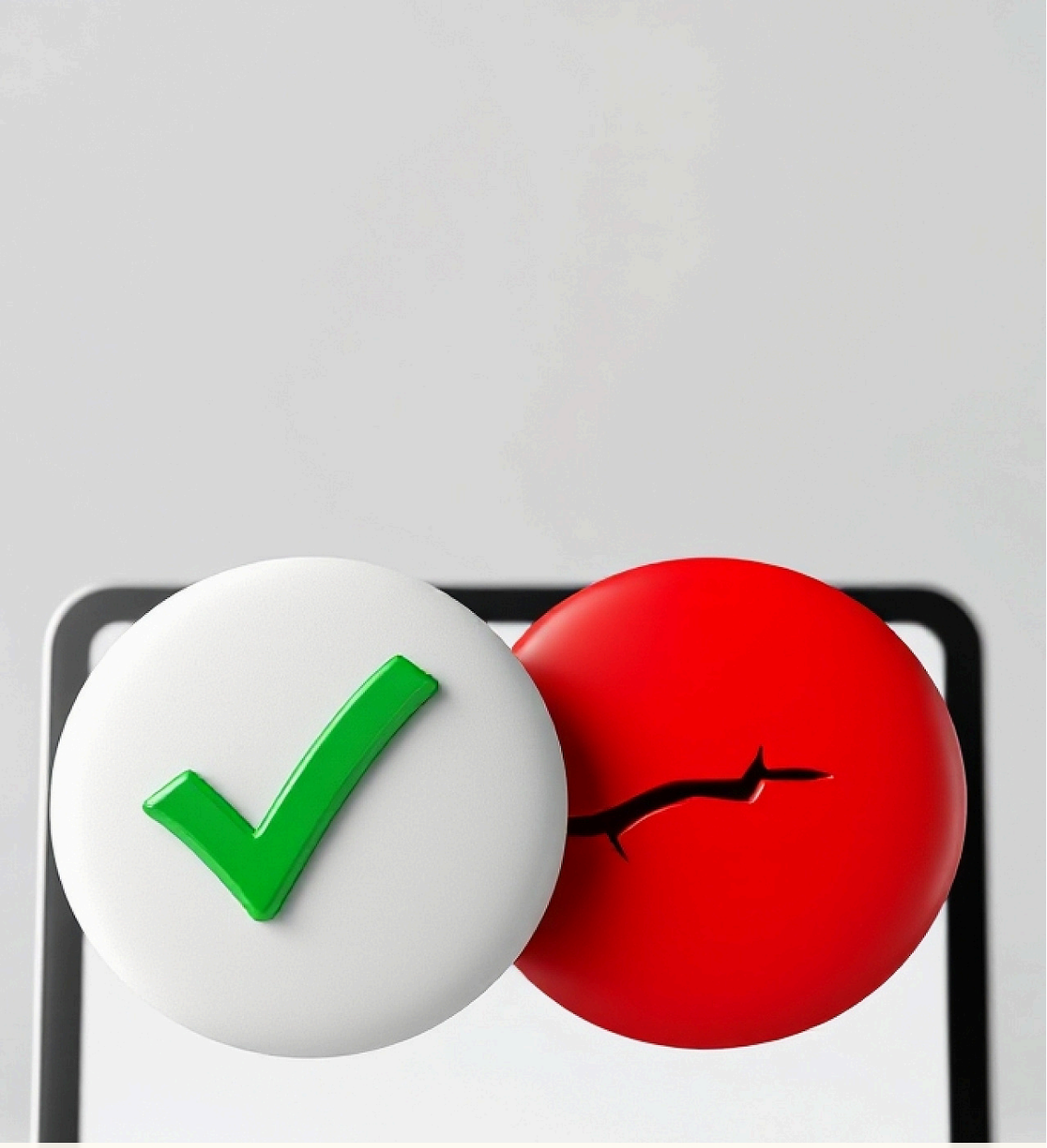
Error handling means deciding what to do when a problem is detected. The program does not just stop; it responds in a planned way. For instance, if a file cannot be opened, the program might log the problem and continue with default settings. Error handling turns a potential crash into a controlled response, keeping the program running or exiting cleanly.



When a problem is detected, two good responses are to tell the user what went wrong or to follow an alternative path. Both keep the program useful instead of failing silently. For example, if a file is missing, the program could display a message saying the file was not found, or it could create a new file with default content. Either way, the user is informed or the program continues.



A common beginner mistake is assuming everything will always work and skipping checks. This turns small real-world problems into runtime errors. For example, a program that always tries to open a file without checking might crash if the file is missing. By adding checks, you can prevent such errors and make the program more robust. Always consider what could go wrong.



Another common mistake is calling every problem a runtime error. Recall that writing mistakes and runtime errors belong to different phases. A writing mistake is caught before running, while a runtime error happens during execution. Confusing them can lead to looking for the wrong cause. Always identify whether the error occurred while writing or while running.



A runtime error happens during execution when an invalid situation occurs, even with correctly written code. Remember to think ahead, add checks, and handle errors so the program can explain the problem or take another path. By planning for the unexpected, you make your programs more reliable and user-friendly. This is the key to managing runtime errors.