How can artificial intelligence find an error in a program?



www.yzokul.com.tr

How can artificial intelligence find an error in a program?



Artificial intelligence can find an error in a program by comparing your code and its error message with patterns it learned from many programs, then suggesting likely problems. You give it the code and the message, and it guesses where the mistake probably is. The short answer is that AI uses learned patterns to spot common mistakes. It does not run your program; it only reads and compares.



A program error is something that stops the program from working or makes it give a wrong result. When an error happens, the program often stops and shows a red message on the screen. For example, a program that adds two numbers might crash if you give it a word instead of a number. The error is the problem that prevents the correct result.



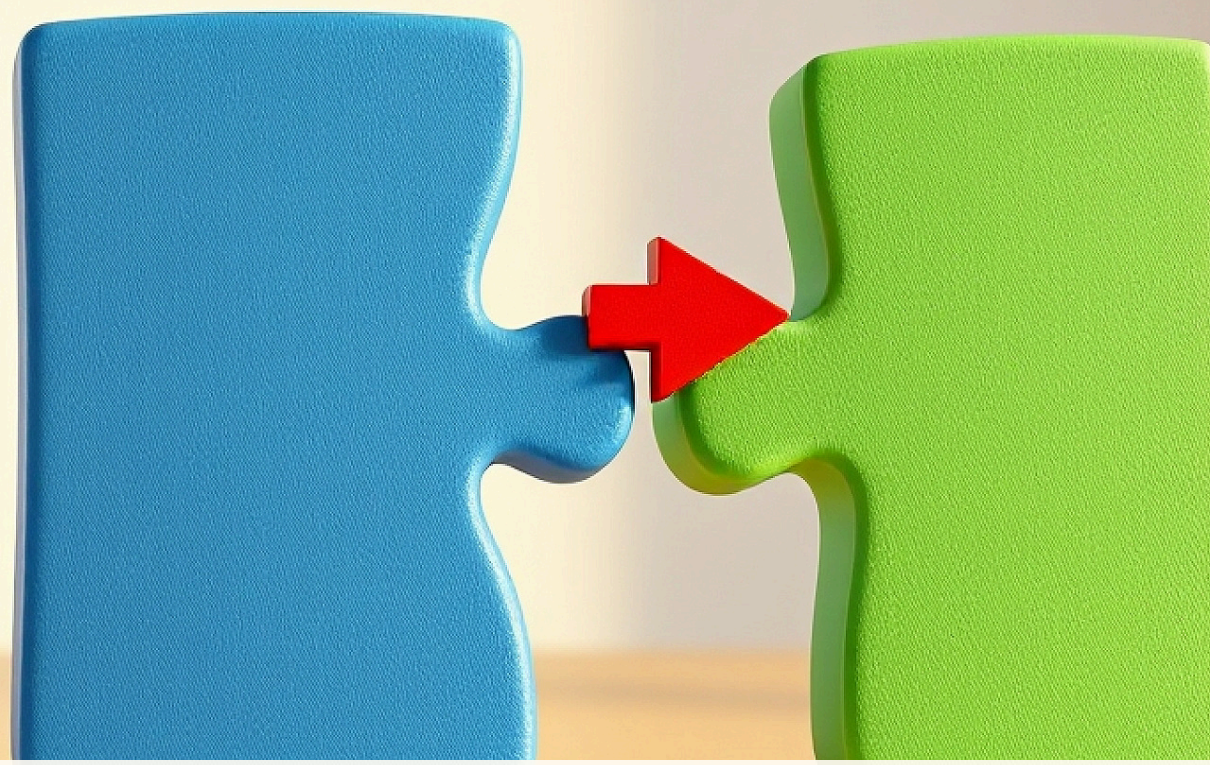
An error message is a short text that a program prints when it fails. It usually says what went wrong and where in the code the problem happened. For example, it might say that a name is not defined or that a list index is out of range. This message is a clue that both you and the AI can read to find the cause.



Artificial intelligence has read a huge number of programs and their errors before you ever use it. From this, it learned common patterns, such as which mistakes often cause which messages. For example, it learned that a certain message often appears when a variable name is misspelled. These learned patterns help it guess what is wrong in your code.



The AI looks at two things at once: your code and the error message. It matches them against its learned patterns and guesses where the problem probably is. For example, if the message says a name is not defined, it searches your code for that name. By combining both clues, it narrows down the likely location of the error.



One common pattern is a wrong variable name. A variable is a name that holds a value, and if you spell it differently in two places, the program cannot find it. The AI spots that the names do not match and suggests the correct spelling. For example, if you write total in one line and totl in another, it points out the mismatch.



Another pattern is a missing check. The code uses something without first checking that it is safe to use. For example, it might try to divide by a number without checking that the number is not zero. The AI suggests adding a check before that step, such as testing whether the number is zero before dividing.



A third pattern is incorrect order of operations. Steps happen in the wrong order, so a value is used before it is ready. For example, you might print a result before you calculate it. The AI points out the order that would work, such as calculating first and then printing. This helps you see why the value was wrong.



Consider a small example where the program tries to take the first item of an empty list. A list is a collection of items, and an empty list has nothing in it. The AI explains that there is nothing to take, which causes the error. It suggests checking that the list is not empty before taking the first item.



The AI can be wrong because it only sees what you show it, not everything the program does when it runs. It does not watch the program execute or see the actual data. Missing information can lead it to a wrong diagnosis. For example, it might blame a line of code that is actually fine because the real problem is elsewhere.



Real programs depend on inputs, data, and settings that the AI may not know. For example, a program might work with small numbers but fail with very large ones. Without these hidden operating conditions, the AI's guess about the cause can be off. It might suggest a fix that does not address the real problem.



Always try the AI's suggestion on the real program. Change the code as suggested, run the program, and see whether the error actually goes away. For example, if the AI says to add a check for an empty list, add it and test with an empty list. Testing is the only way to know if the fix works.



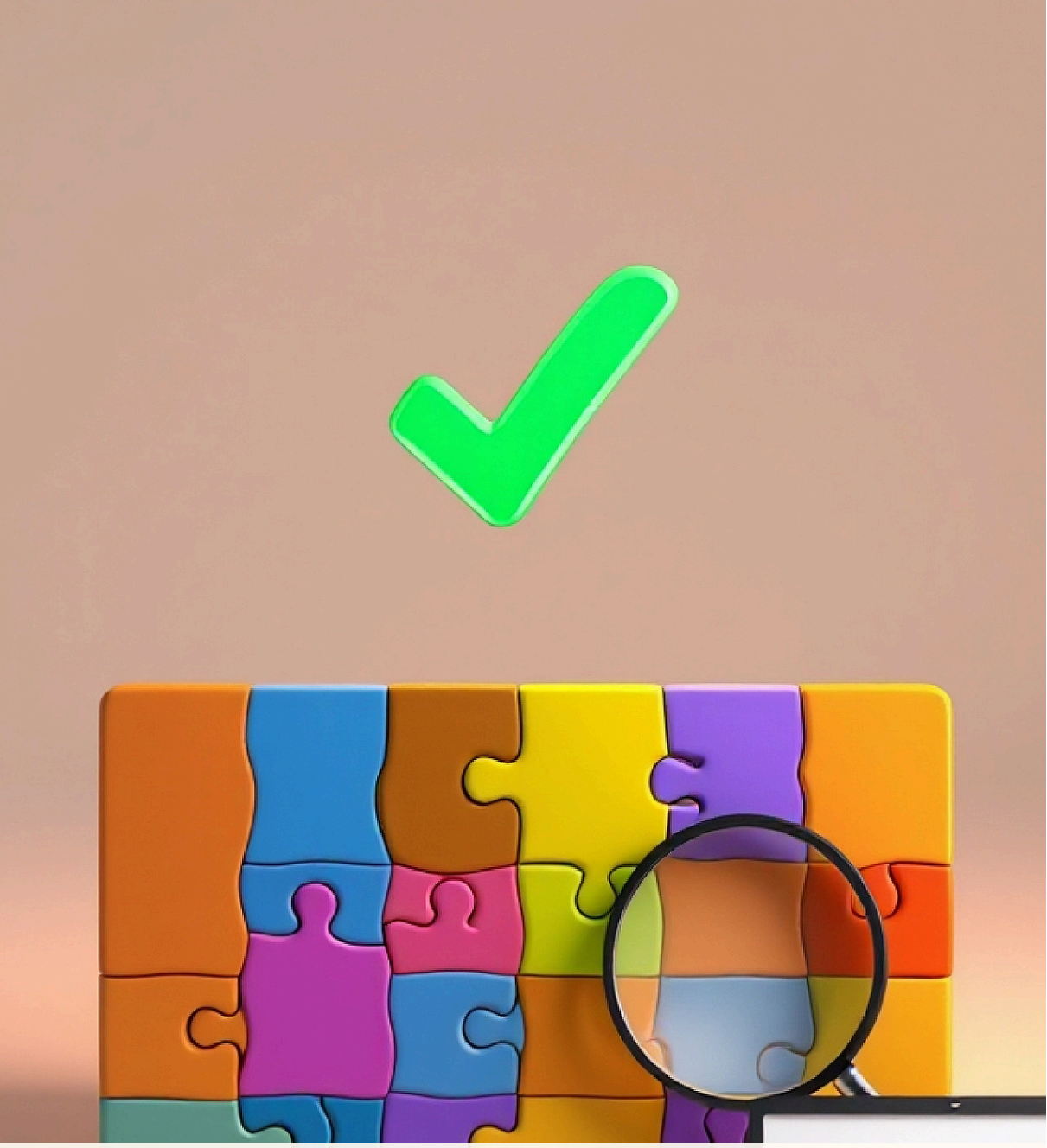
A clear, confident explanation from the AI can still be wrong. The explanation may sound convincing, but it is only a guess based on patterns. Only running the real program shows whether the fix truly works. Do not accept an explanation as proof; always test the change and observe the result.



A typical beginner mistake is accepting the AI's answer without testing it. You read a convincing explanation, change the code, and assume it is fixed. But when you run the program again, the same error appears. The picture shows the error message returning because the fix was never tested on the real program.



Another mistake is giving the AI too little information. If you show only part of the code or no error message, its suggestion is a guess, not a diagnosis. For example, without the full error message, it cannot know which line failed. Always provide the complete code and the exact error message for a better guess.



Artificial intelligence finds likely errors by matching your code and error messages to patterns it learned from many programs, but it can be wrong. It can suggest problems like a wrong variable name, a missing check, or incorrect order of operations. Always test its suggestion on the real program before trusting it. A convincing explanation alone is not enough.