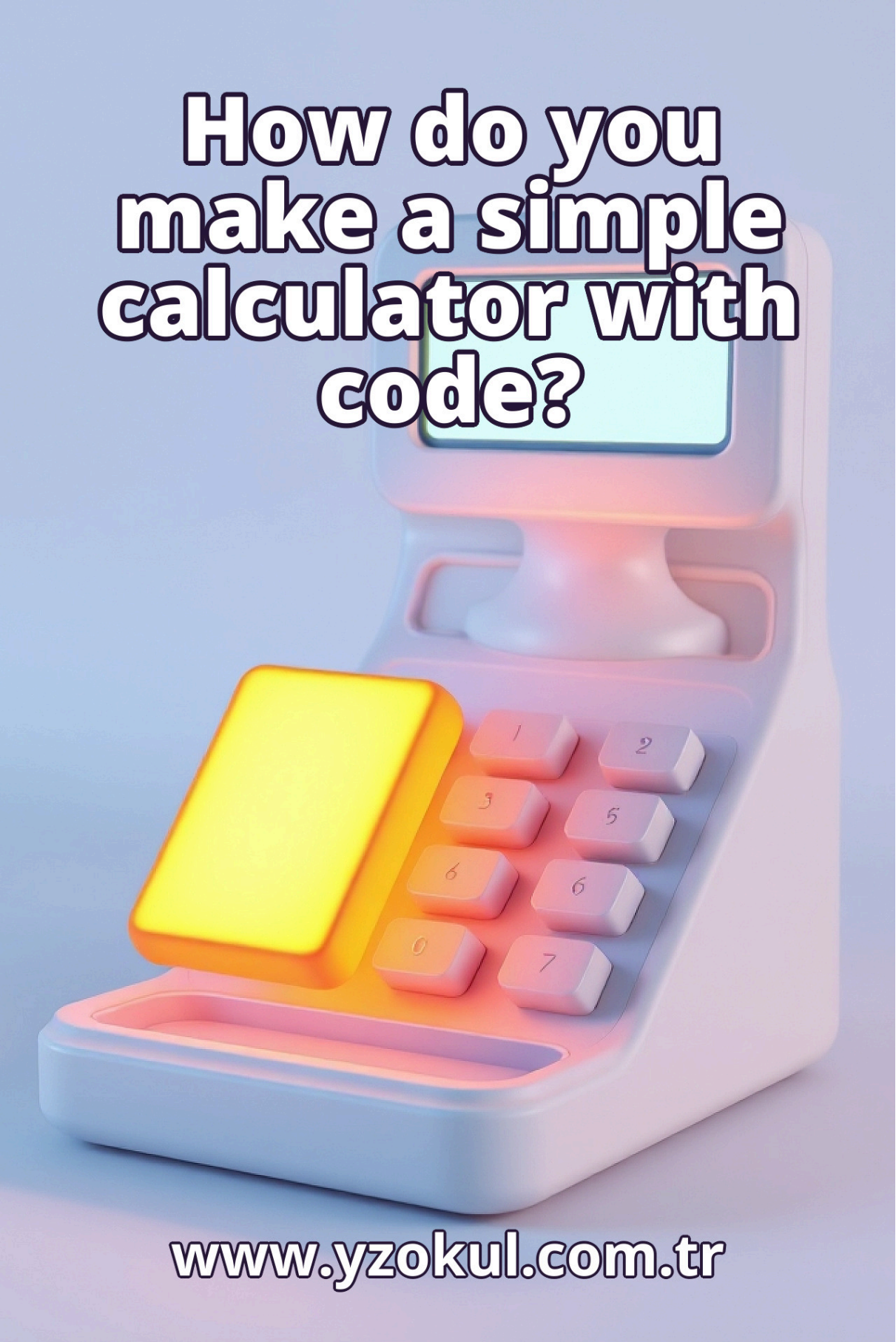
How do you make a simple calculator with code?

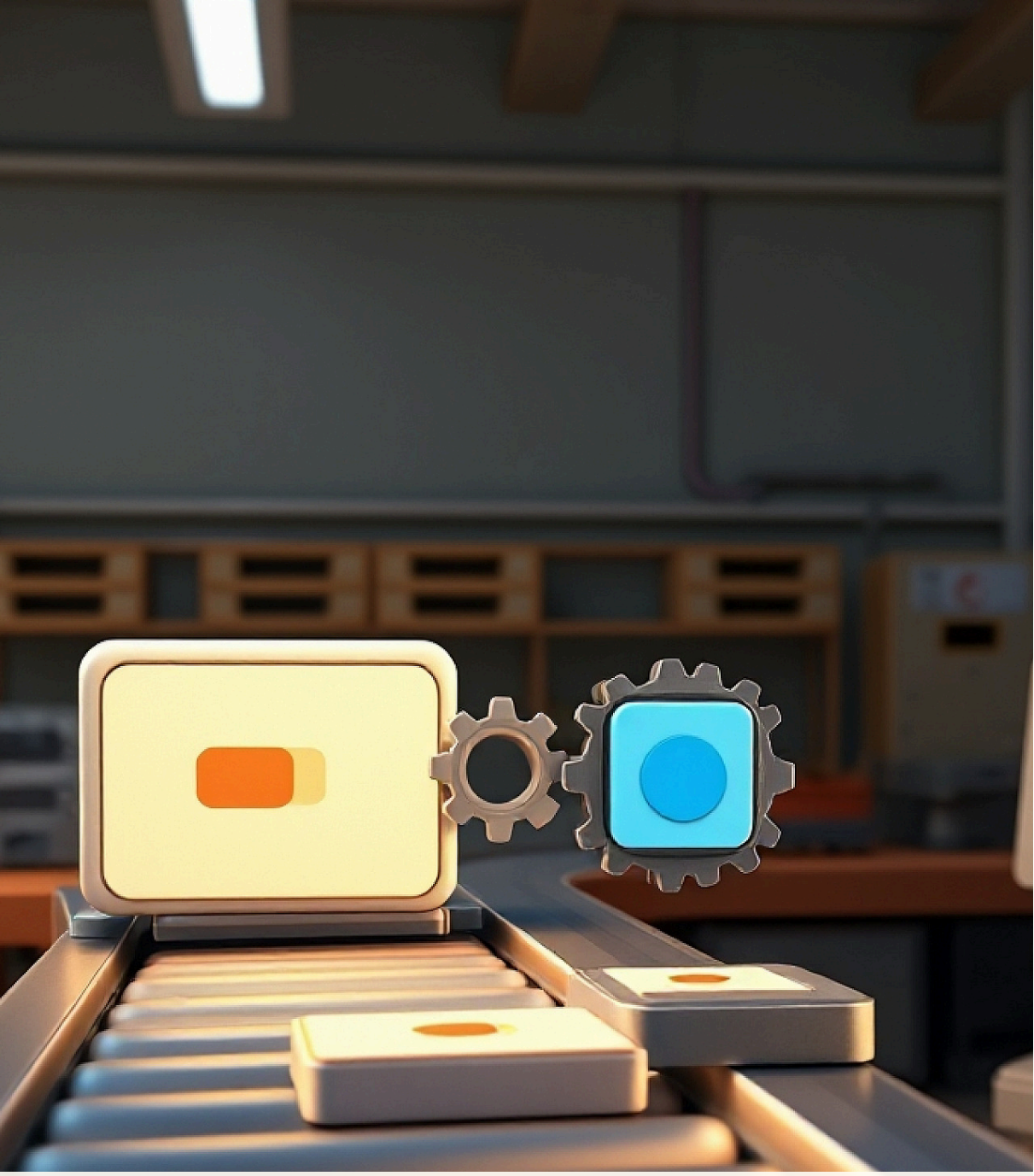


www.yzokul.com.tr

How do you make a simple calculator with code?



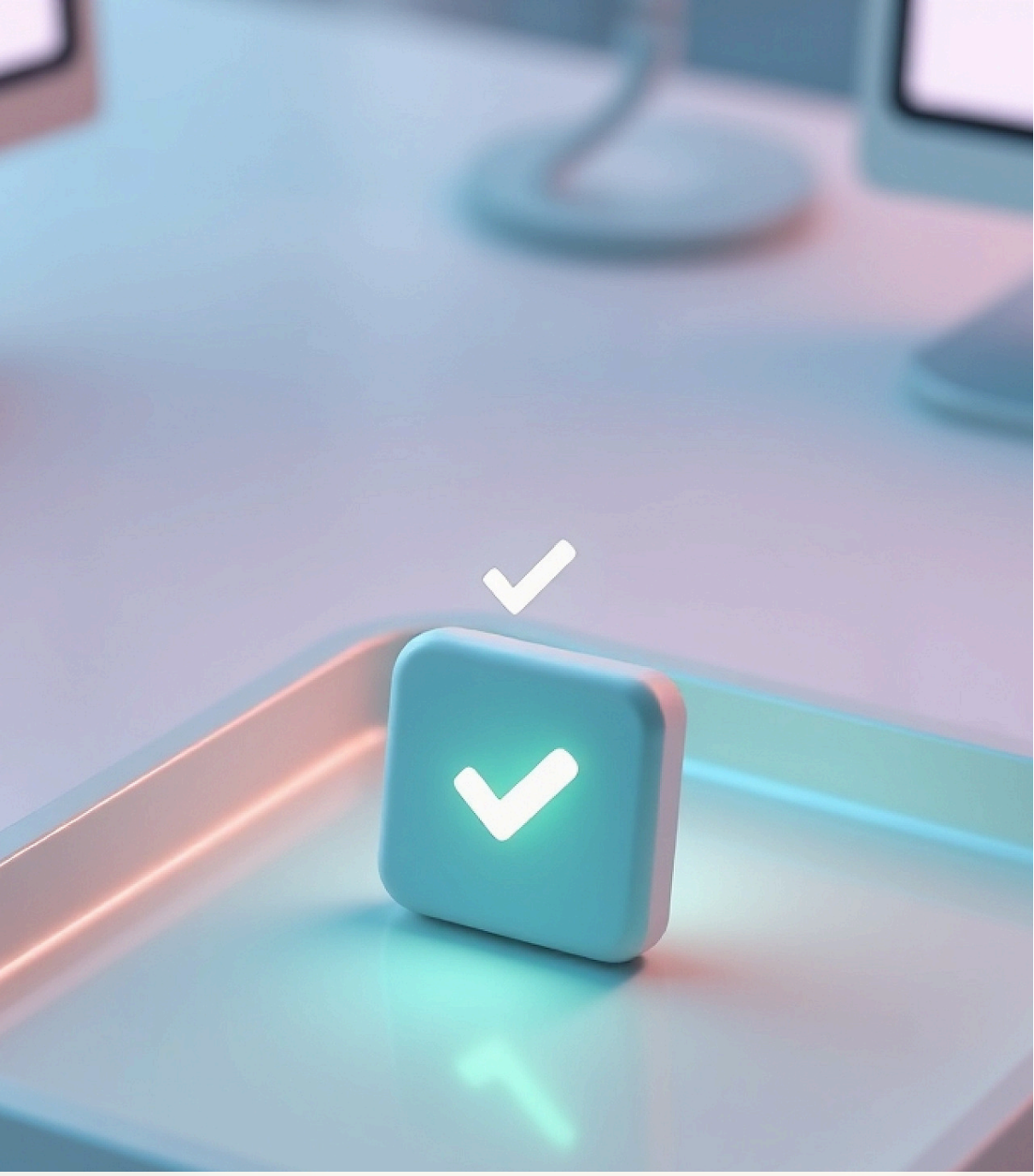
A simple calculator is a program that takes two numbers and an operation, then shows the result. The short answer is that code stores the inputs, picks the right math operation, and prints the answer. For example, if you enter 5 and 3 and choose add, the program shows 8. This book explains each part so you can build one yourself, even with no prior coding experience.



Every calculator does the same basic job: ask for two numbers, ask which operation to perform, do the math, and show the result. This plan is the foundation for all the steps that follow. For example, the program might ask for a first number, then a second, then whether to add, subtract, multiply, or divide. Keep this simple flow in mind as you build each piece.



Pick one beginner-friendly language to start with, such as Python. The same ideas—variables, conditions, functions, and error handling—work in any language, so what you learn here transfers. All examples in this book use Python because its code is short and easy to read. For instance, printing a message in Python takes just one line: print("Hello").



Write and run the smallest possible program to confirm your tools work. In Python, you can print a greeting with a single line. This removes fear before you add real logic. For example, type print("Calculator ready") and run it; you should see the message appear. Once that works, you are ready to build the calculator.



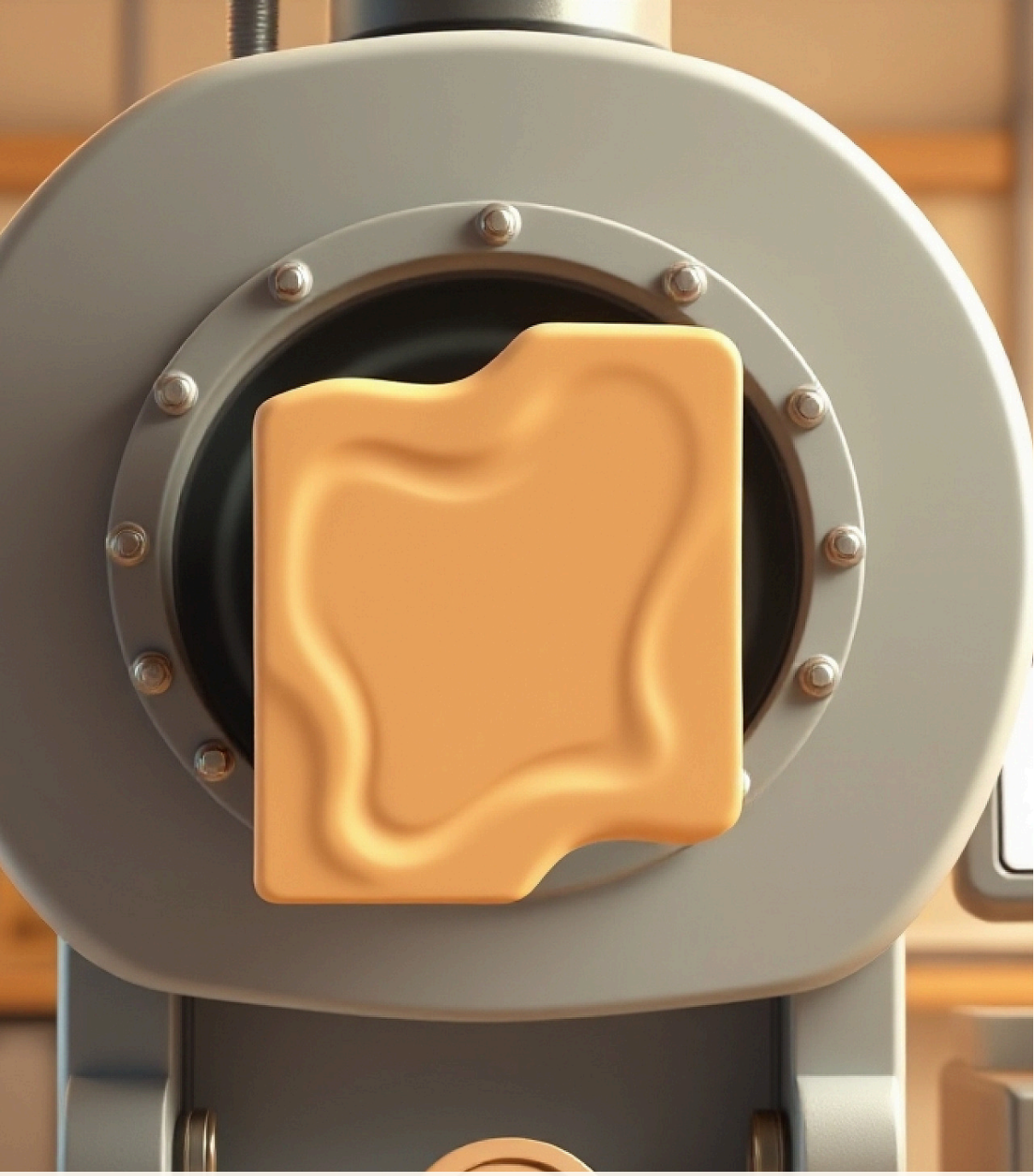
A variable is a named box that stores a value, like a number or text. You will use two variables to hold the two numbers the user types. For example, you can create a variable called first_number and later put a value in it. Variables let you refer to the same value multiple times without retyping it.



Use an input command to ask the user for the first number and the second number. The program pauses and waits until the user types something and presses Enter. For example, in Python you write first = input("First number: "). The text inside the parentheses is the prompt shown to the user. The typed answer is stored in the variable first.



Typed input arrives as text, not as a number. This is why '2' plus '3' could give '23' instead of 5 if you are not careful. For example, if you store the input in a variable and try to add it directly to another input, the program joins the text together. You must convert the text to a real number before doing math.



Convert the text input into a real number using a conversion function. In Python, the function int() turns text into a whole number, and float() turns text into a decimal number. For example, number = int(input("Enter a number: ")) stores a real number you can calculate with. Now the program can actually do math with the values.



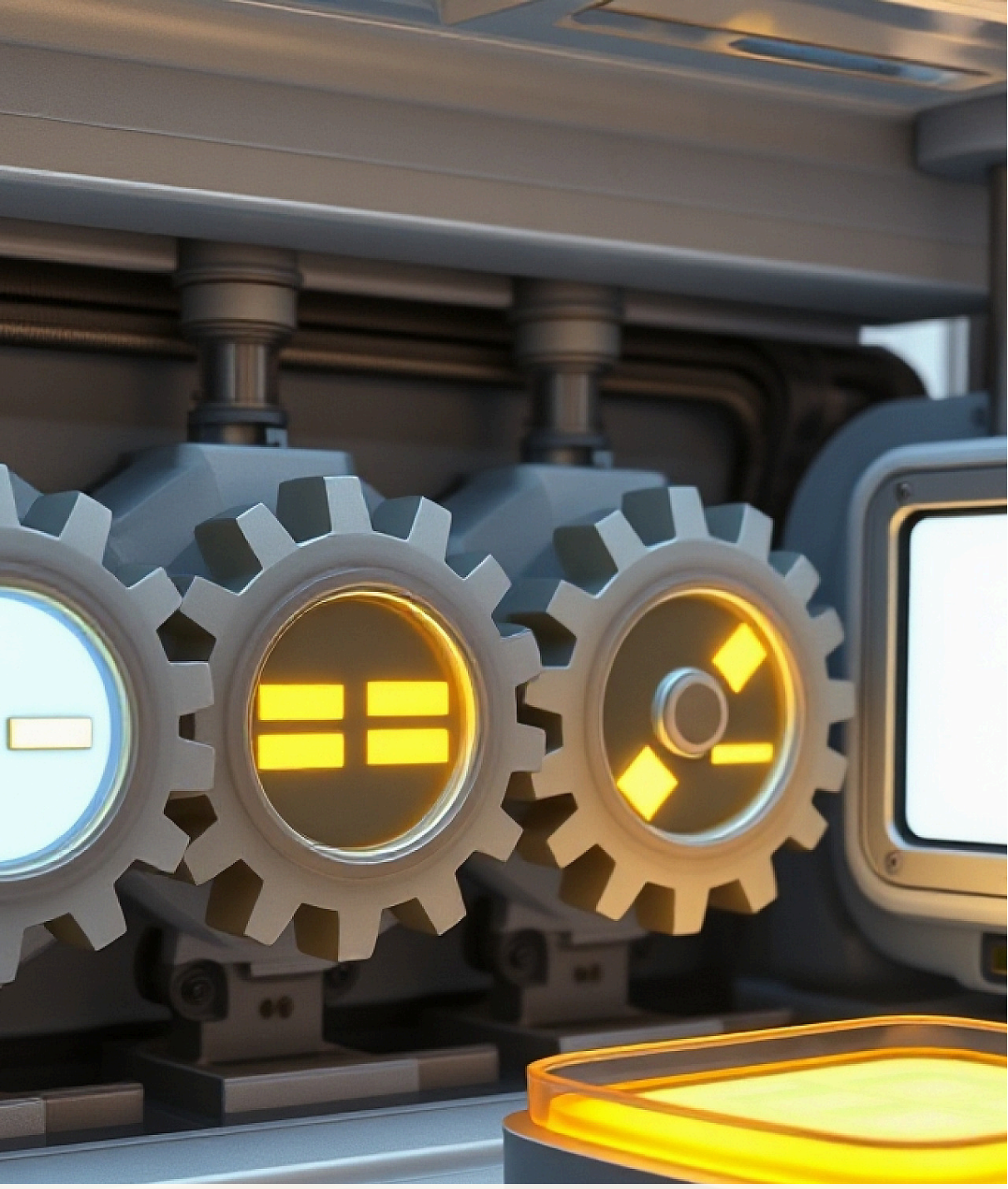
Ask the user which operation to perform, such as add, subtract, multiply, or divide. Store the choice in a variable so the program can decide what to do next. For example, op = input("Choose operation (+, -, *, /): "). The user types one of the symbols, and that symbol is saved for the next part.



An if statement checks a condition and runs code only when it is true. This is how the program picks the right operation from the user's choice. For example, if op == "+": runs the addition code. You can add more conditions with elif for other operations, and a final else for anything unexpected.



Write the branches that add or subtract the two numbers and store the result. Each branch handles exactly one operation. For example, if op == "+": result = a + b; elif op == "-": result = a - b. Here a and b are the converted numbers, and result holds the answer.



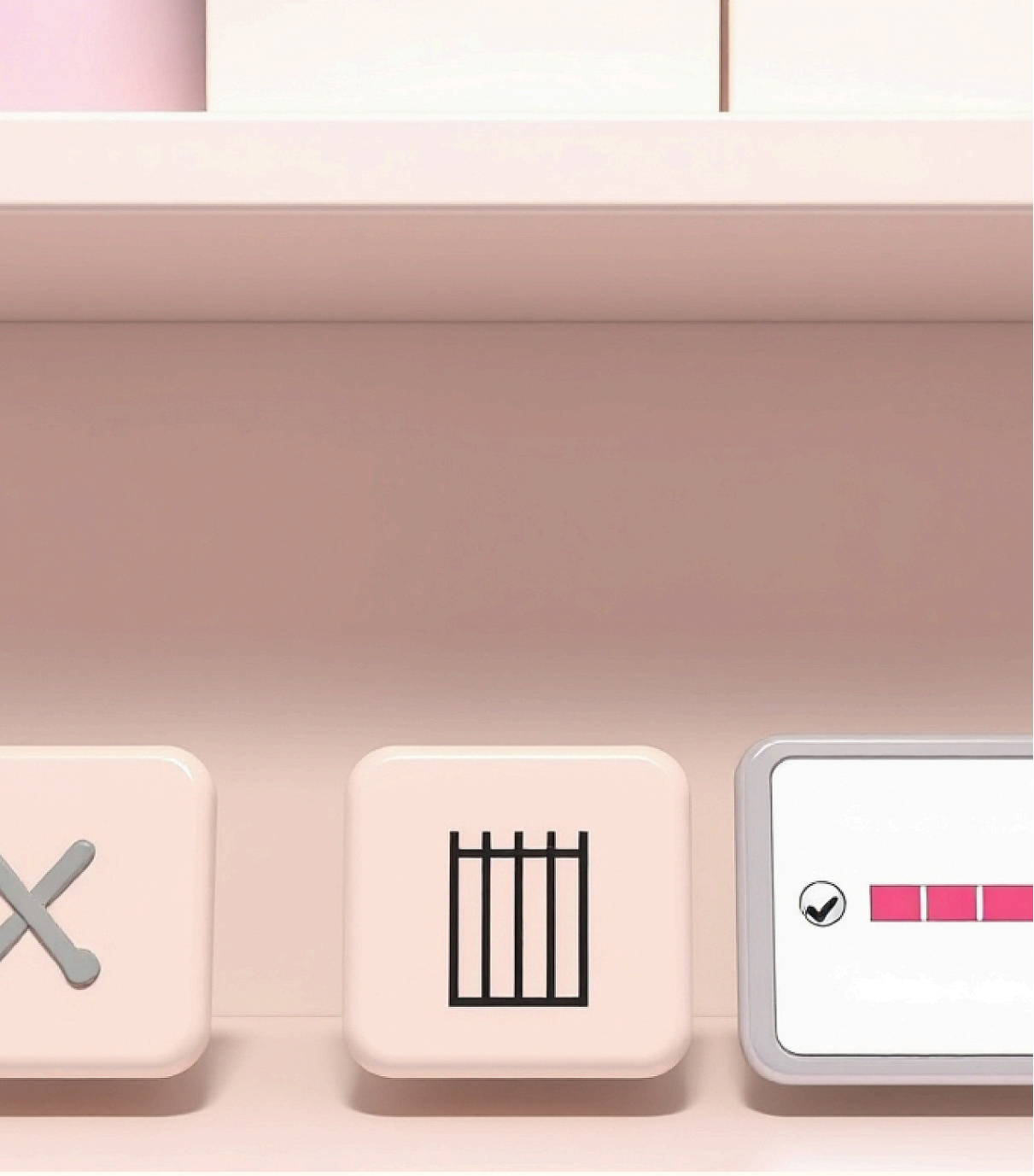
Add the branches for multiplication and division, completing the four basic operations. The result is stored in the same result variable. For example, elif op == "*": result = a * b; elif op == "/": result = a / b. Now the calculator can handle all four common math operations.



Dividing by zero is an error, so the program must check the divisor before dividing. If the divisor is zero, show a clear message instead of crashing. For example, before dividing, check if b == 0: print("Cannot divide by zero"). Only divide when b is not zero. This is called error handling.



Print the stored result in a clear, readable way, such as 'Result: 8'. The output area is where the user finally sees the answer. For example, use print("Result:", result). Make sure the message is easy to understand so the user knows what the number means.



Watch for common beginner mistakes: forgetting to convert input, putting conditions in the wrong order, and missing the zero check. Each mistake has a simple fix. For example, always convert input with int() or float() before math; check division by zero before dividing; and order your if and elif branches so the correct one runs.



Recall the full path: get input, convert it, choose an operation with conditions, check for errors, and show the result. The key ideas to remember are variables, conditions, functions, and error handling. For example, a complete calculator combines all these steps into a short program that works reliably. With this foundation, you can build more features later.