What is programming?



www.yzokul.com.tr

What is programming?



Programming means writing a set of steps that a computer can follow to solve a problem. It is not magic or a special talent; it is the skill of giving clear, ordered instructions. You start with a problem, decide what steps will solve it, and write those steps in a form a computer can run. The computer then carries them out exactly as written.



A computer does exactly what it is told, in the order it is told, and nothing more. It cannot guess what you meant or fill in missing details. If a step is unclear or absent, the computer will not know what to do. So every instruction must be clear and complete, leaving no room for doubt about what should happen next.



Before writing any commands, you must understand the problem clearly. Ask yourself what result you want and what information you already have. For a class grade average, the goal is one number: the average grade. The information you have is the list of grades for each student. Knowing both helps you plan the right steps.



Designing the solution means planning the steps needed to reach the answer before you touch a computer. This plan is the real solution, and it comes before any commands. For the grade average, the plan is simple: take the grades, add them together, and divide by the number of students. Writing this plan first keeps you from guessing later.



Put the planned steps into the right sequence, one after another. The order matters because a computer carries them out exactly as written, from first to last. If you add before you have the grades, or divide before you add, the result will be wrong. So arrange the steps so each one uses what the previous step produced.



Use the class grade average as a simple example to work through. The goal is to find the average grade of a class of students. You have a list of grades, one for each student. The average is a single number that represents the whole class. This example shows how a problem becomes a set of clear steps.



The first step is to collect the grades you have. Without the input data, there is nothing to calculate. In programming, this information is called input. For the class average, the input is the list of grades, such as 80, 90, and 70. Gather all the grades before doing any arithmetic, so the next steps have something to work with.



Add all the grades together to get one total. This is a single, clear step the computer can carry out. For example, if the grades are 80, 90, and 70, the total is 240. Adding is a simple operation, but it must happen before dividing. The total becomes the input for the next step, which is division.



Divide the total by the number of students to get the average. Each step builds on the one before it: the total from adding is now used in the division. For example, 240 divided by 3 students gives an average of 80. This final step produces the answer you wanted, and it depends on the earlier steps being correct.



The steps are written in a programming language, which is a set of words and rules a computer can understand. The language is the tool for expressing your solution, not the solution itself. Many languages exist, and they all let you write clear steps. Choose one that suits your problem and learn its basic rules for writing commands.



The computer runs the program by carrying out the operations according to the rules you wrote. It follows your steps in order and produces the result. For the grade average, it adds the grades, divides by the count, and shows the answer. The computer does not think or judge; it simply executes the instructions you provided.



Test the result by checking whether the answer is correct. Testing is part of programming, not something separate from it. Compare the computer's output with what you expected. For the grade average, you can calculate the average by hand: 240 divided by 3 is 80. If the program shows 80, it works; if not, you need to find and fix the mistake.



A common beginner mistake is jumping straight to writing commands without understanding the problem. This leads to programs that do not solve what was asked. The fix is to design the steps first: know the goal, the input, and the order of operations. Only then should you write any commands in a programming language.



Another common mistake is putting steps in the wrong order, like dividing before adding. The computer will follow your wrong order and give a wrong answer. For the grade average, dividing before adding would use the wrong numbers. The fix is to check the sequence carefully and make sure each step uses the result of the previous one.



Assuming the program works without checking is a mistake. Even simple steps can hide errors. The fix is to test the result and compare it with what you expected. For the grade average, calculate the answer yourself and see if the program matches. If it does not, review your steps and fix the problem before trusting the output.



Programming is understanding a problem, designing steps, writing them in a language, and testing the result. The key thing to remember is that a computer only follows clear steps in order. It cannot guess or fill gaps. By planning first, writing in sequence, and testing, you can solve problems with a computer reliably.