What is a comment line for?





www.yzokul.com.tr

What is a comment line for?



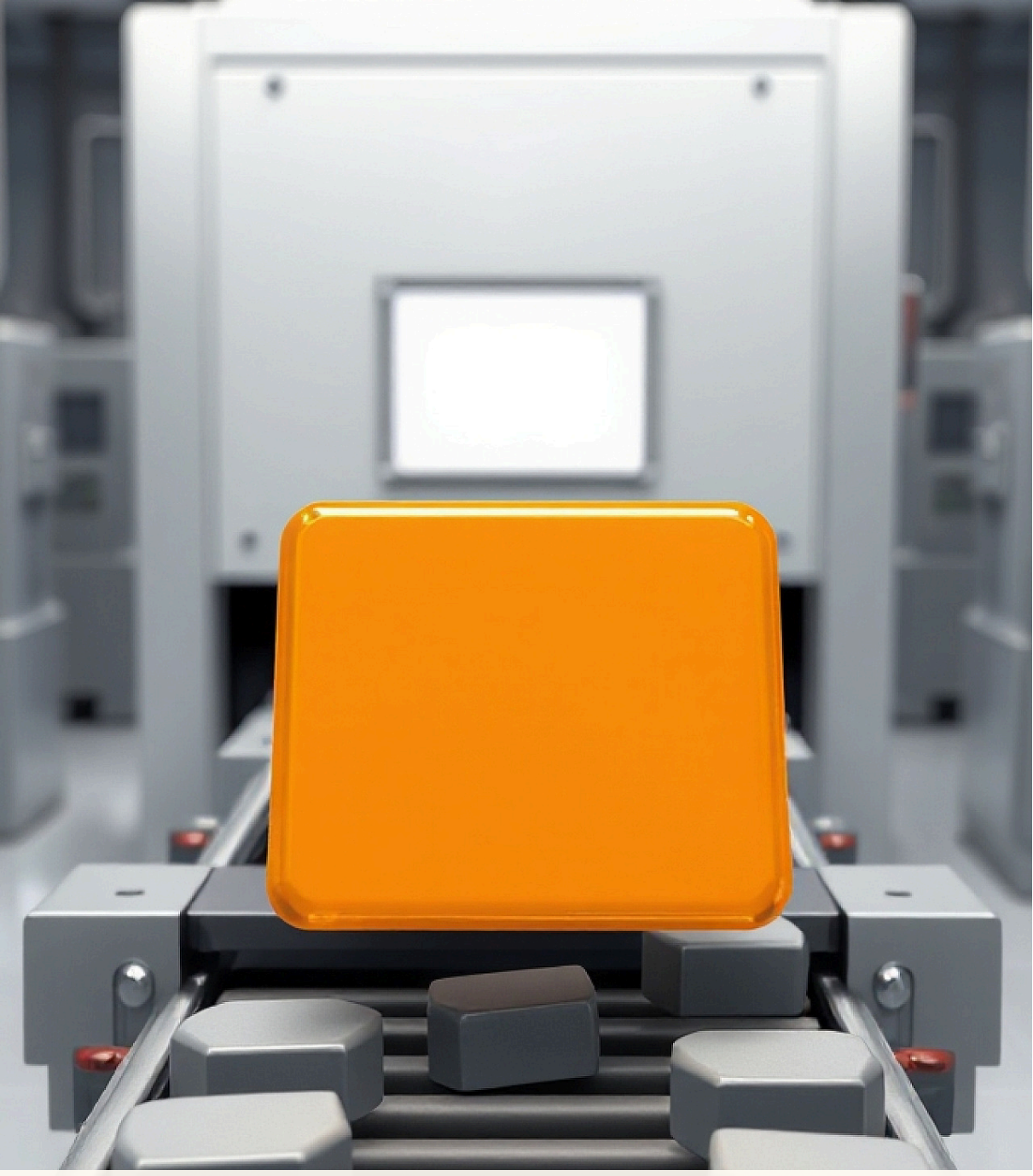
A comment line is a note written inside the code for people to read. It gives an explanation or extra information. It is not normally run as a command, so the computer skips it when the program runs. Think of it as a message from the writer to anyone who reads the code later. Its main job is to help people understand the code better.



Code is written for a computer to execute, but it is also read by people later. Those people might be other programmers or even you in the future. Comments are the place where those readers get extra explanation that is not in the code itself. Without comments, readers have to guess the meaning. Comments make the code easier to understand for everyone.



Picture a comment as a note stuck inside the code, like a sticky note on a page. It sits next to the code but speaks to humans, not to the machine. The computer ignores it, but people can read it. You can write a comment on its own line or at the end of a code line. It is there to give context and clarity.



When the program runs, a comment is normally skipped. The computer does not treat it as an instruction to do something. It is not a command that makes the program act. Instead, it is just text that the computer passes over. This means you can write comments freely without changing how the program works. They are only for human readers.



A comment can explain why a decision was made in the code. The reason behind a choice is often the most useful thing to write down. For example, you might choose a certain method because it is faster or safer. A comment can record that reason so others understand. Without it, someone might wonder why you did it that way.



A comment can also describe the purpose of a complex operation. When a part of the code is tricky, a comment helps a reader grasp what it is trying to achieve. It can summarize the goal in simple words. This is especially helpful when the code itself is long or hard to follow. The comment gives the big picture.



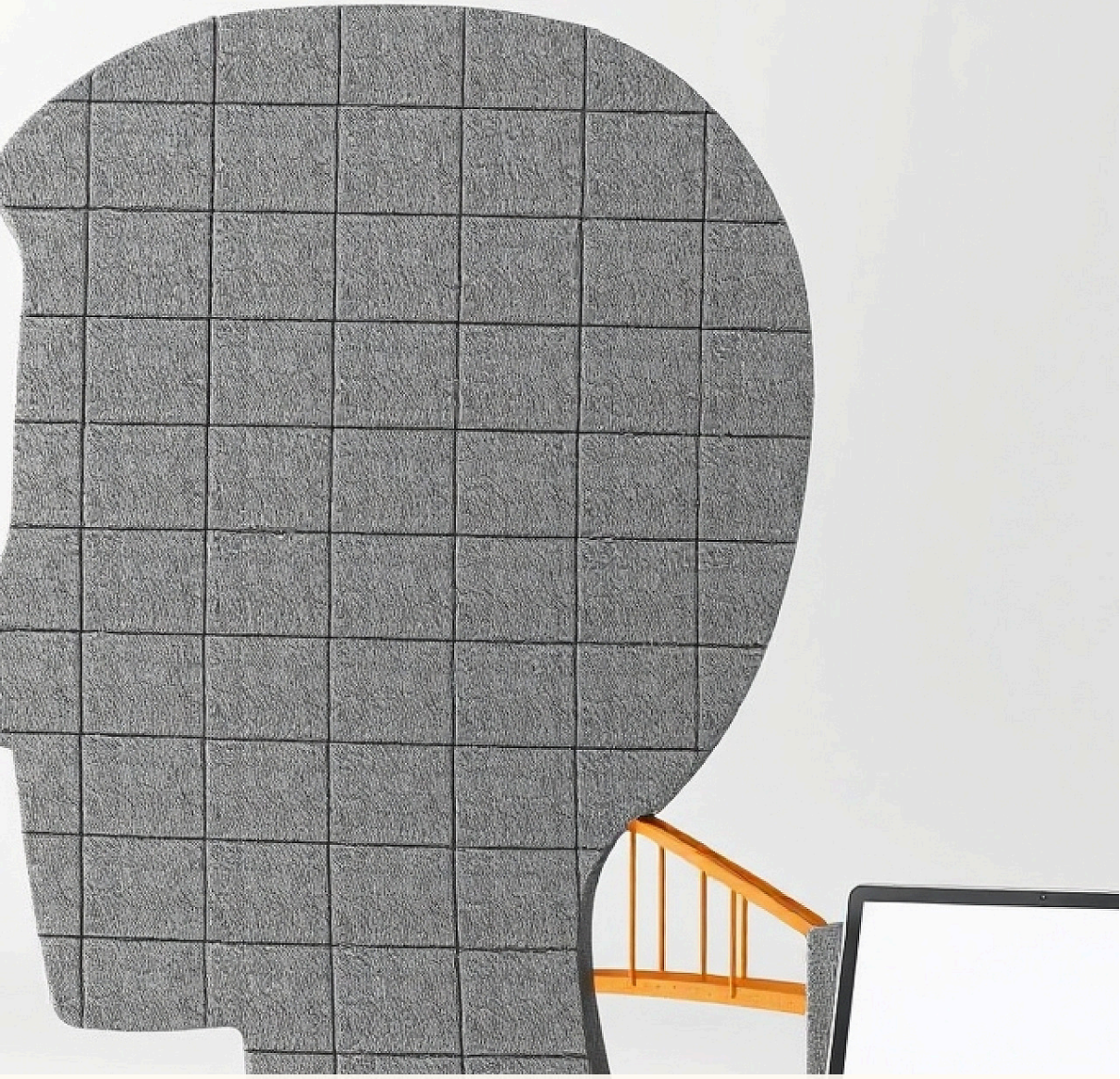
Consider a simple example where a comment explains why a particular limit is used. Suppose you set a maximum of 100 items. A comment might say that this limit is based on memory constraints. The comment tells the reader the reason behind the number. Without it, the number might seem arbitrary. The comment makes the choice clear.



Repeating what the code already says clearly is not always useful. A comment that just restates the obvious adds little value. For instance, writing a comment that says increment the counter above a line that clearly increments the counter is redundant. Good comments focus on information that is not already obvious from reading the code. They add new insight.



Good comments explain the reasons and assumptions that are not easy to understand. These are the things a reader cannot see from the code alone. For example, you might assume that a certain input is always positive. A comment can state that assumption. Or you might have a reason for using a specific algorithm. Comments capture that hidden knowledge.



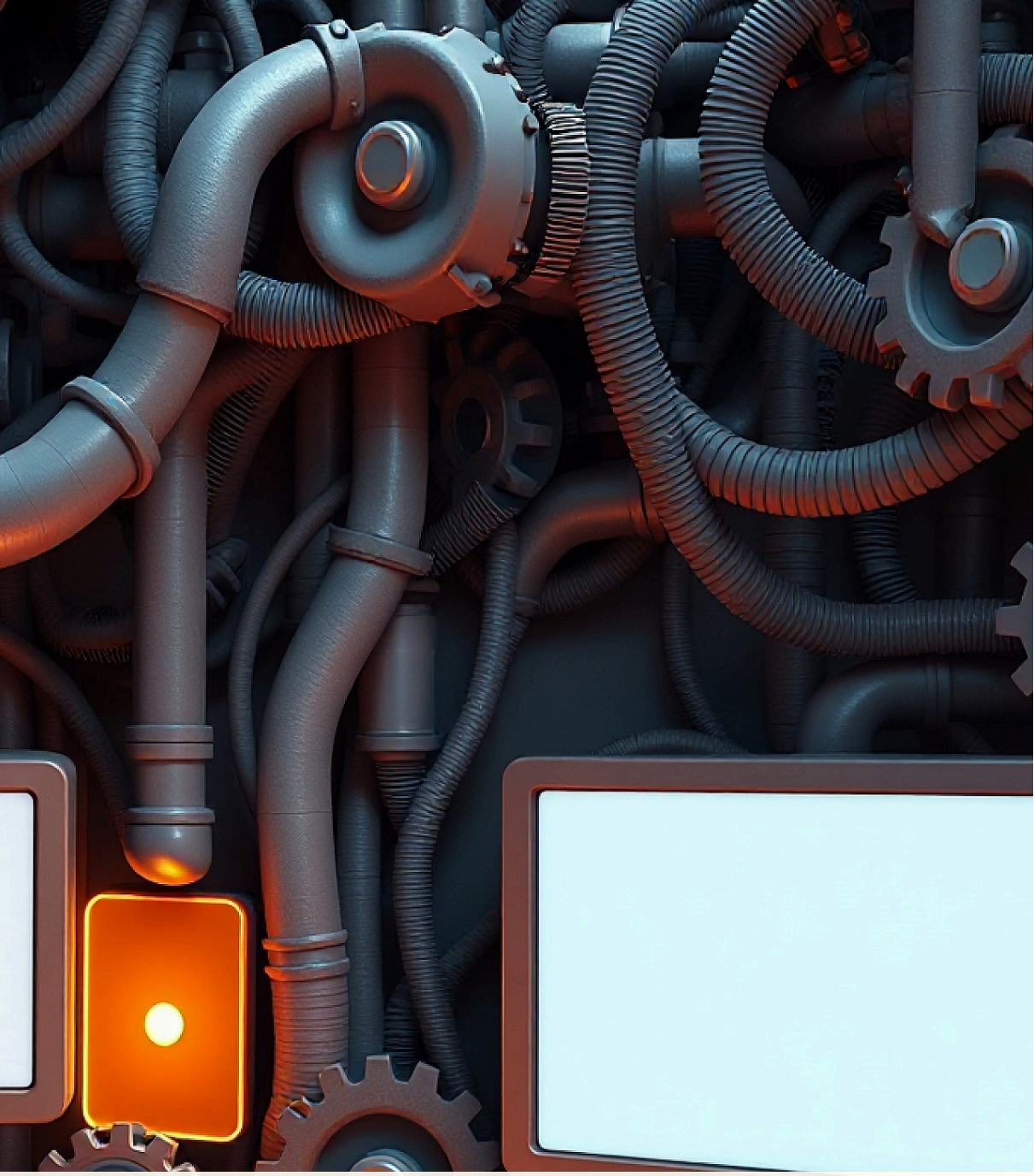
Some knowledge lives only in the writer's head. Comments pass that hidden knowledge on to the next person. When you write code, you make many decisions that are not written down. Later, someone else may read the code and wonder why. A comment can share your thoughts and prevent confusion. It bridges the gap between writer and reader.



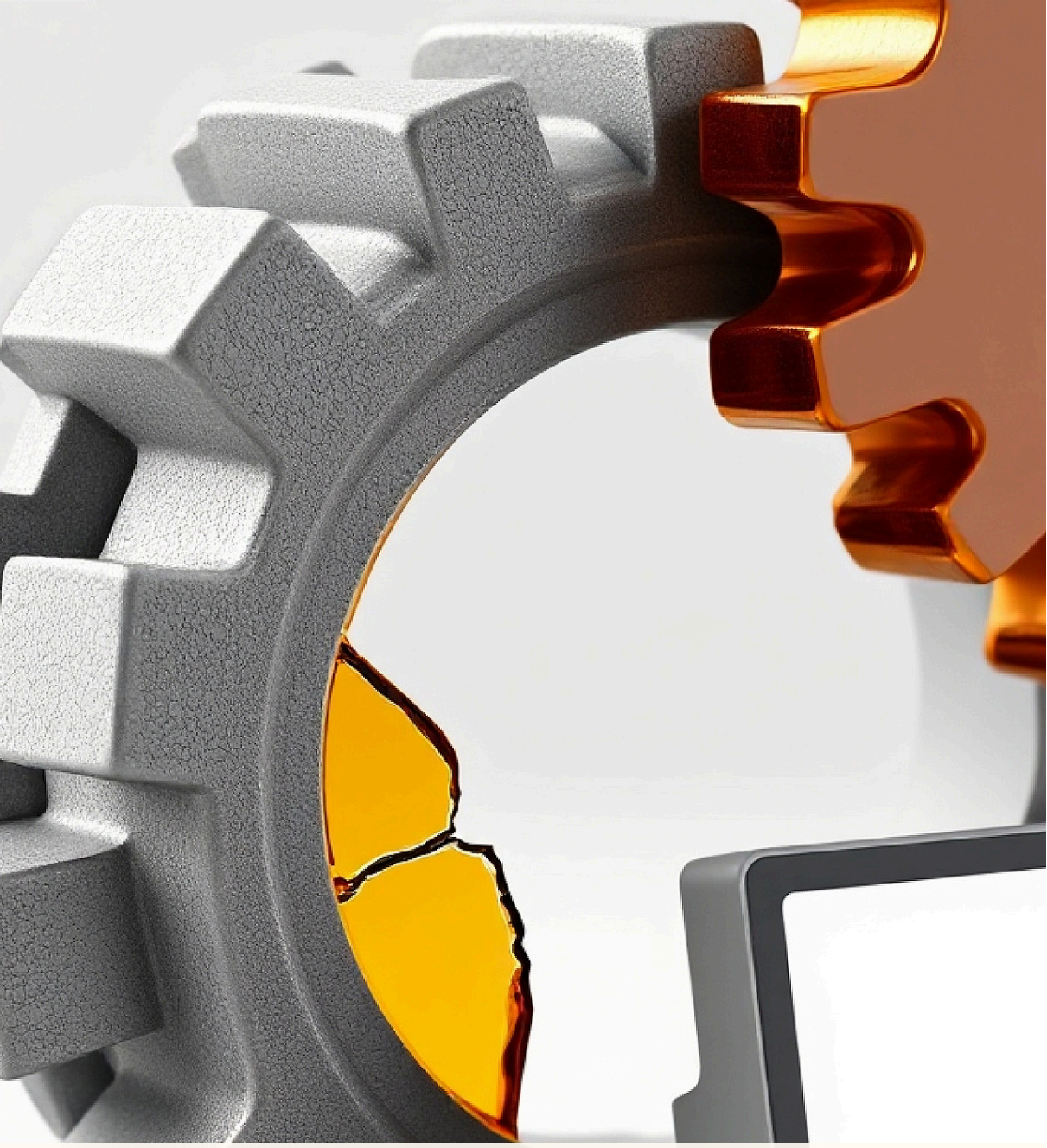
A comment and a command are different. A command makes the program act, like printing a message or calculating a value. A comment only explains; it does not cause any action. The computer runs commands but skips comments. So, when you write code, you use commands to do things and comments to explain. Both are important for different reasons.



A common mistake is writing comments that only repeat the code. For example, a comment that says set x to 5 above the line x = 5 is not helpful. The fix is to focus on reasons and assumptions instead. Ask yourself why you are doing something and what assumptions you are making. Then write that in the comment. This makes the comment valuable.



The opposite mistake is leaving a tricky decision with no explanation at all. If you write complex code without any comment, the next reader may struggle to understand. A short note about the reason would help. For instance, if you use a unusual approach to solve a problem, explain why you chose it. This saves time and reduces frustration for others.



A comment that no longer matches the code can mislead people. If you change the code but forget to update the comment, it may give wrong information. This is worse than having no comment because it can cause misunderstandings. Comments should be kept honest and up to date. Whenever you modify code, check that nearby comments are still accurate.



Comments help other people and also your own future self. When you return to code after months, you may forget why you did something. A clear comment reminds you. Other people reading your code also benefit from your explanations. Anyone reading the code later gains from a clear explanation. So, writing comments is a good habit for everyone.



To sum up, a comment line explains things to people and is not normally run. Good comments give the reasons and assumptions that are hard to see. They help readers understand why decisions were made and what complex operations do. Avoid repeating the obvious and keep comments accurate. Use comments to share your knowledge and make code easier to understand.