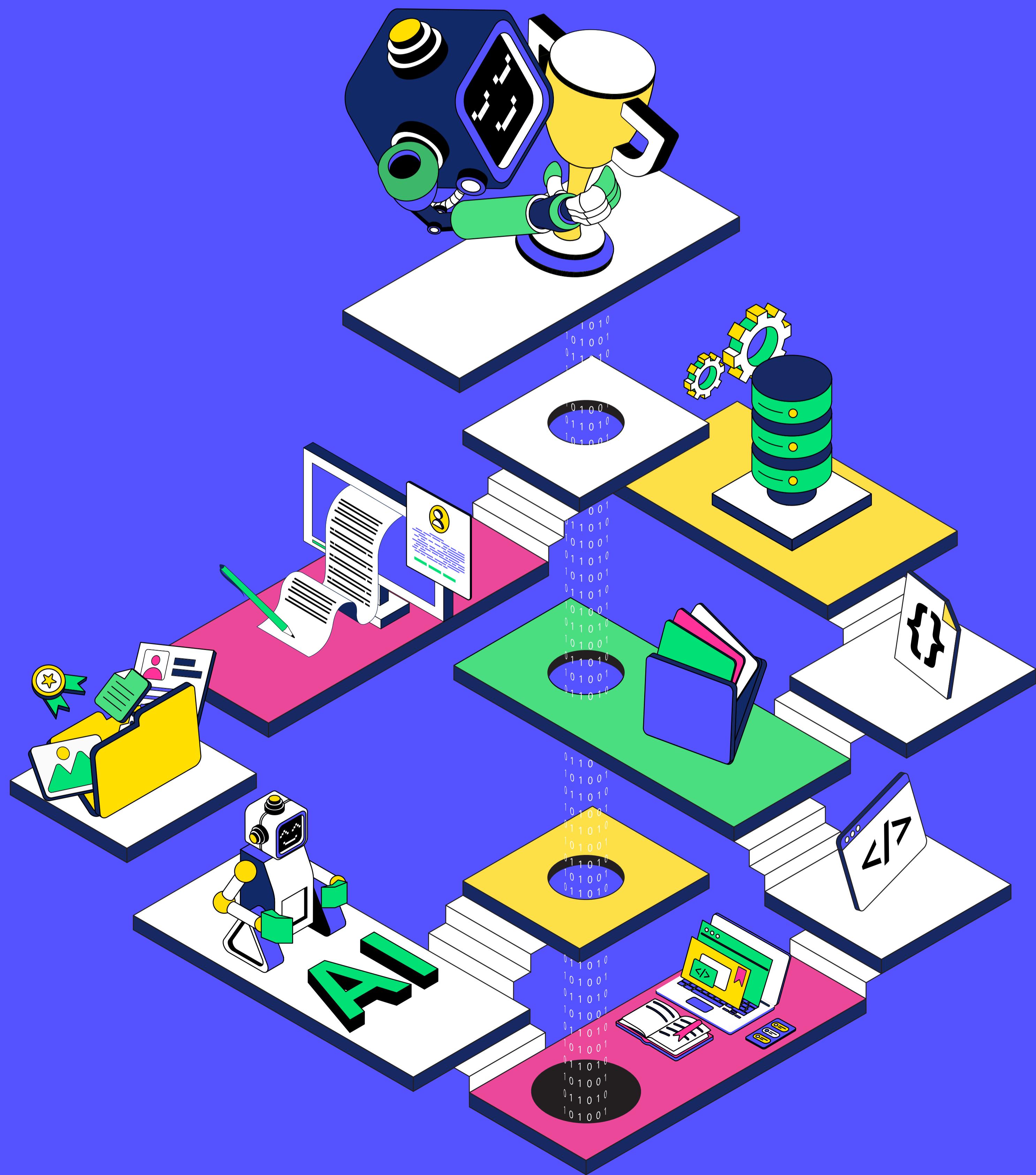


8 WEEK

ROADMAP TO ACE YOUR SOFTWARE ENGINEER INTERVIEW

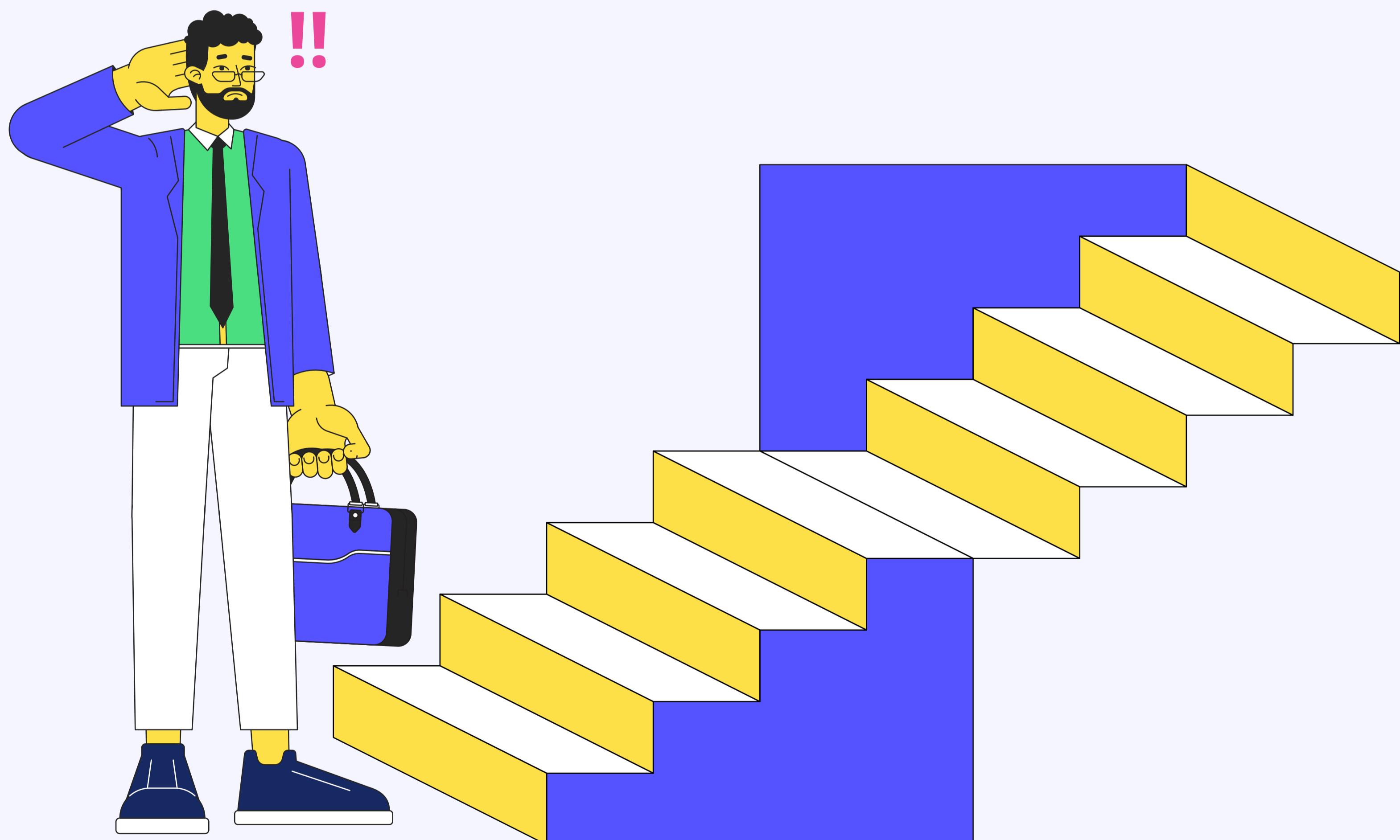


WHY EVERY DEV NEEDS A STRATEGIC INTERVIEW PREP PLAN

Out of every **100** software engineer applicants at top companies, only **5** are typically selected for a preliminary screening. Then out of every **5** applicants who make it to the final interview round, only **1** will get an offer.

To put it simply: software engineer roles are competitive.

A strong performance in the technical interview loop is critical for landing a job as a software engineer. Going to an interview unprepared can mean losing out on the first big step in your career.



That's why we created this 8-week roadmap: to help developers brush up on basics, refresh their interview skills, and walk into their loops with confidence.

Each week will break down the technical topics to review along with questions to practice.

Before we share a few essential tips for efficient prep, here is an overview of the 8-week roadmap:

8-WEEK INTERVIEW PREP ROADMAP

Week 1

Brush up on programming basics

Weeks 2 & 3

Review data structures & algorithms

Week 4

Practice with data structures & algorithms

Weeks 5 & 6

Test your skills with real-world coding problems

Weeks 7 & 8

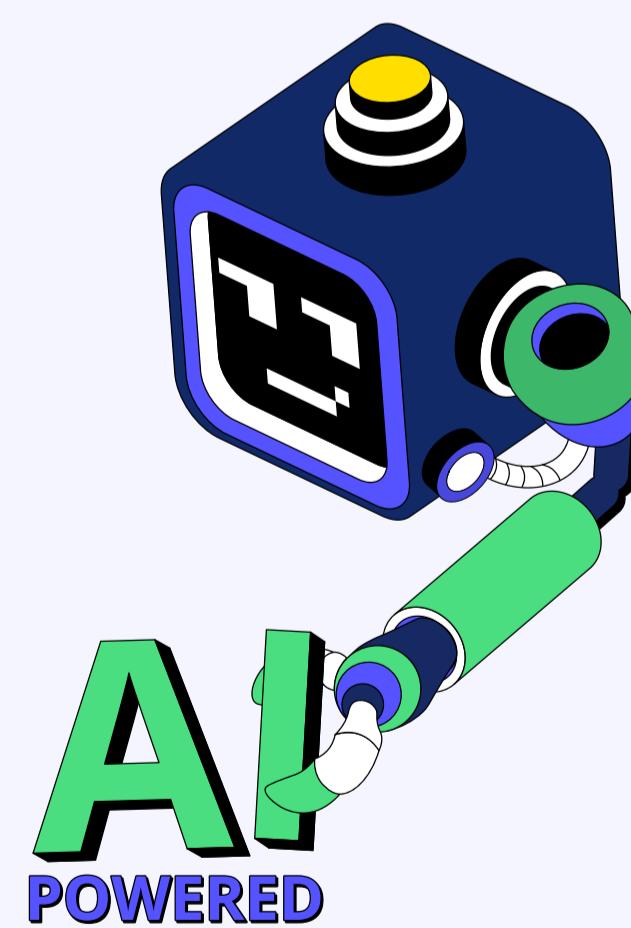
Design interview & behavioral interview

Note that each week will be broken down in detail later.

EDUCATIVE IS YOUR HACK TO FASTER INTERVIEW PREP

Join over **2.6 million developers** leveling up their careers with Educative. Our interactive learning platform offers hands-on experiences designed to help you build and practice in-demand skills.

2.6 MILLION DEVELOPERS



Our resources have helped developers get hired at Google, Microsoft, Amazon, Meta, and Apple. The strategies you will learn are developed by MAANG hiring managers to help candidates navigate interview loops at top companies.

> [Everything you need for every interview, all in one place.](#)

WHAT'S THE FASTEST PATH TO INTERVIEW SUCCESS?

The job market for software developers is as competitive as ever. That means standing out from other candidates in the interview loop can be a challenge.

In such a tough job market, the only thing you fully control is how you prepare.

Here's an example timeline to help you prepare comprehensively, but efficiently.

CODING INTERVIEW

(60-90 MINS)

SYSTEM DESIGN INTERVIEW

(45-60 MINS)

BEHAVIORAL INTERVIEW

(45-60 MINS)

A structured prep plan gives you a framework for studying crucial topics, while helping you efficiently budget your time. On the day of your interview, you don't want to be caught off guard by a question you didn't anticipate. Knowing you've done everything you can to prepare brings peace of mind when you're in the hot seat.

> Disclaimer: You may need more or less time; this is just a frame of reference. You should consider our suggested review topics and example problems in the context of your needs.

TAILORED INTERVIEW PREP PLANS: SMARTER PRACTICE, FASTER RESULTS

Educative's AI-powered, adaptive Interview Prep Plans help you prepare more efficiently for every phase of your technical interview loop. Powered by PAL (Personalized Adaptive Learning), these plans adjust to your strengths and skills gaps — so you can skip irrelevant topics and focus only what you need to know for your unique interview.

How it works:

- **Personalized roadmap:** Enter your target role, company, and interview date to receive a custom prep plan.
- **Adaptive practice:** PAL adjusts your practice based on performance — if you ace Linked Lists, you'll move on. And if you need more time to nail Dynamic Programming, you'll get it.
- **AI Mock Interviews:** Test your skills with realistic mock interviews and personalized feedback.



With a tailored plan, you can avoid wasting time on topics you've already mastered, and reach interview readiness faster - focusing on your gaps, your strengths, and your goals.

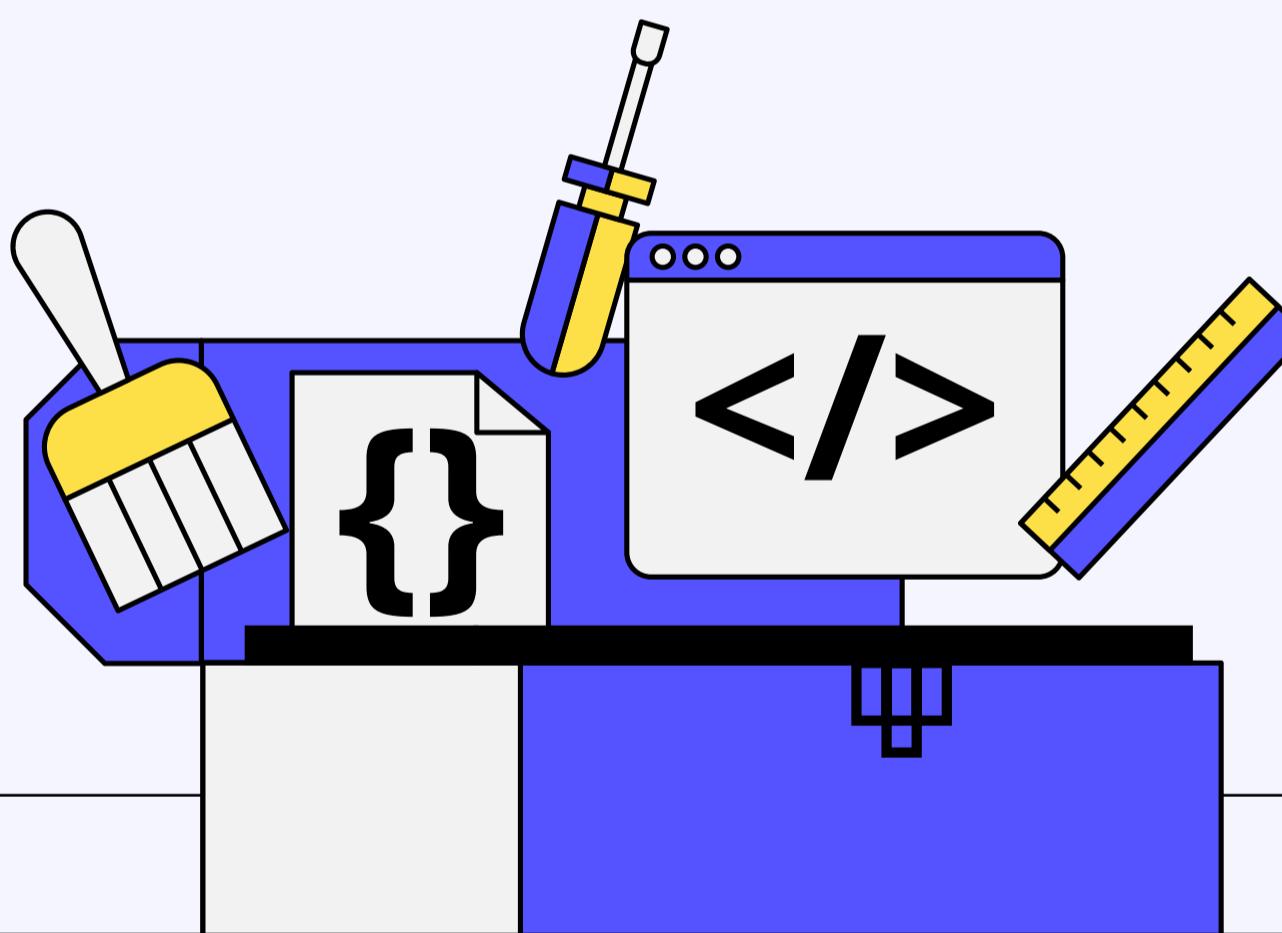
THE COMPLETE 8-WEEK ROADMAP

This framework is ideal for new developers and students preparing to land their first role, but it's great for established programmers who want to keep their skills sharp, too.

WEEK 1

Brush up on the basics of your chosen language

Many technical interviews start with easy questions to raise the candidate's confidence. Don't get tripped up by something simple at the very beginning!



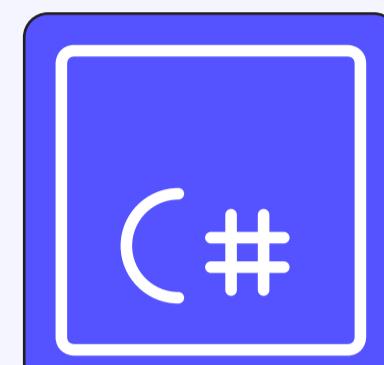
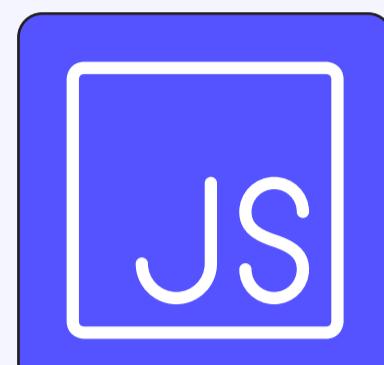
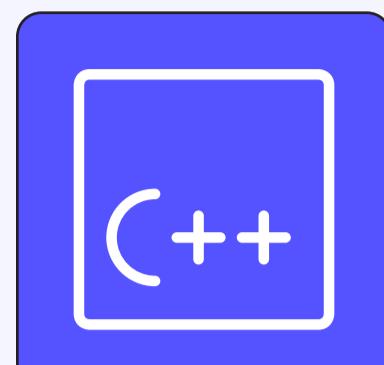
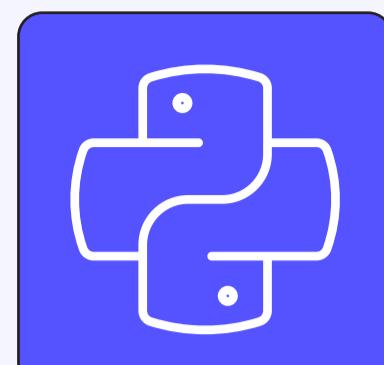
Your fundamentals should be automatic. You don't waste any time during interviews remembering how to perform basic tasks, like tokenizing a string or handling asynchronous calls to an API.

Examples of topics to cover:

- Splitting strings
- Parsing CSV or text files
- Declaring and using 2D arrays
- Reading and writing to and from files
- Processing command line arguments

Brush up on your fundamentals

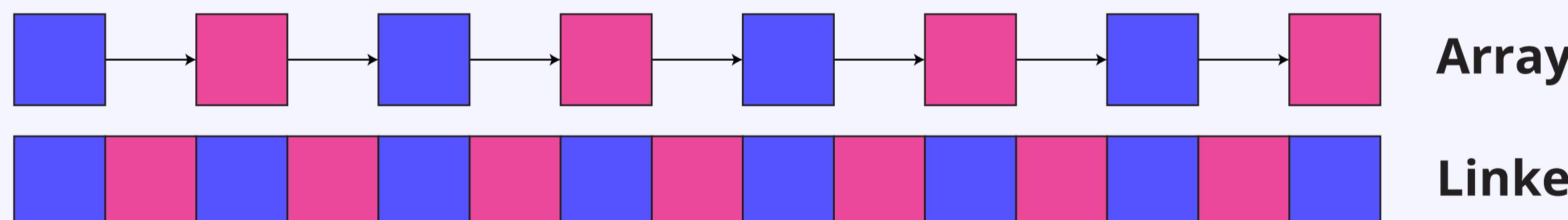
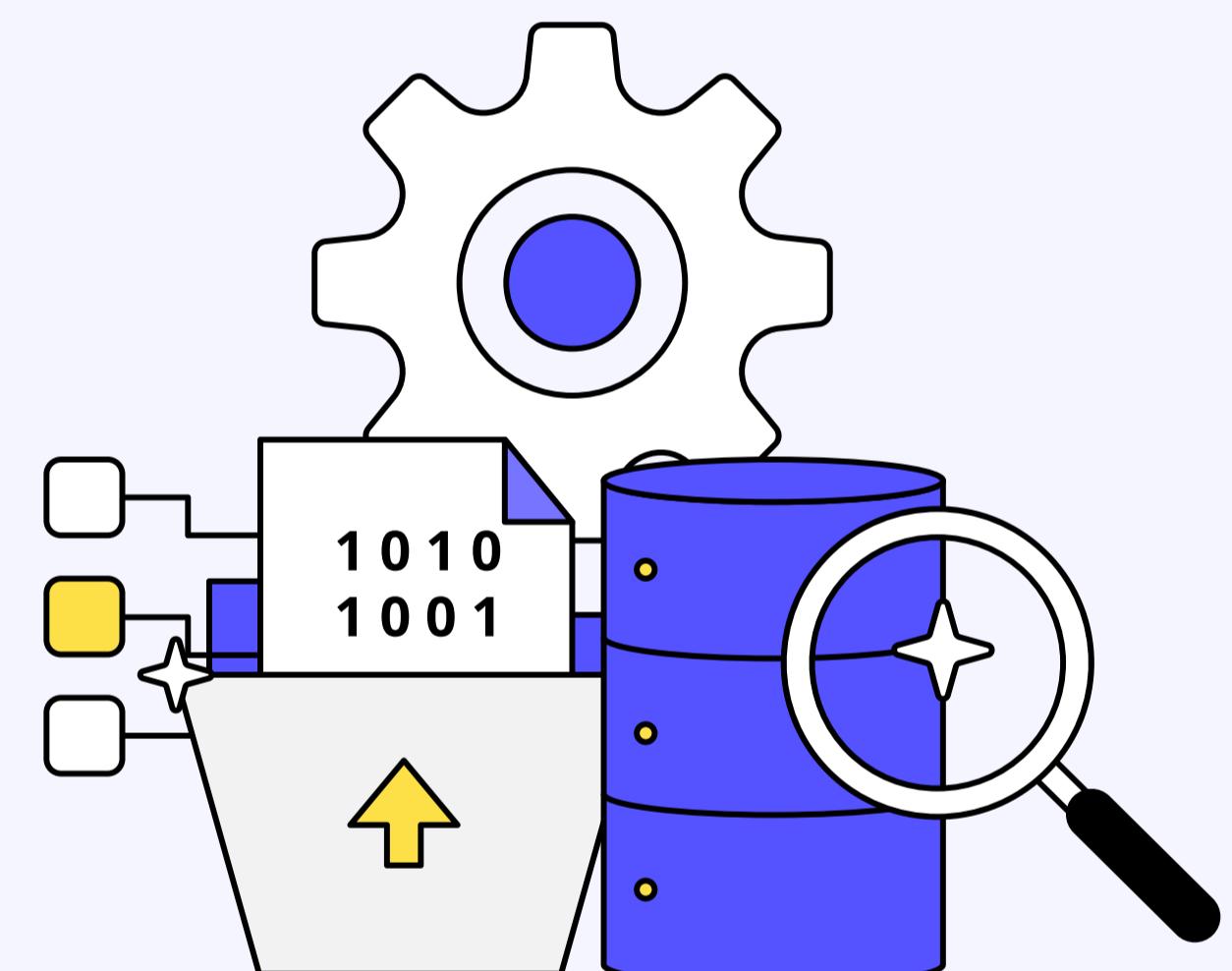
Master the basics.



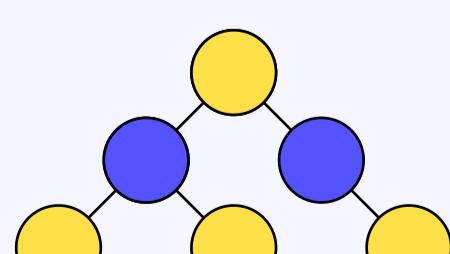
WEEKS 2 AND 3

Review data structures and algorithms

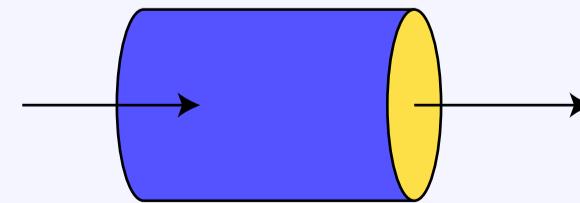
Data structures and algorithms (DSA) are the core of most high-level computer science concepts. These concepts are essential in coding interviews. Before diving straight into example questions, we recommend you study up on the principles of data structures and algorithms. If you're aiming for a specialized role it is smart to consider what individual concepts are the most relevant to the work you'll be doing.



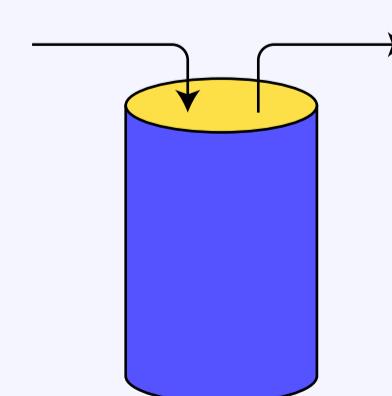
For frontend developers, that could mean studying up on trees and common traversal approaches (**relevant to DOM manipulation**), as well as graphs algorithms (**relevant to component hierarchies and routing**).



Tree



Queue



Stack

For backend developers, don't miss hash tables (important for indexing and caching), as well as sorting and searching algorithms (important for data retrieval and manipulation). Regardless of your specialization, pay particular attention to Big O notation and other practices for complexity analysis. Understanding Big O notation helps in an intense **technical interview**, but it also teaches you to write programs that are faster and more efficient across the board.

Below is a reference guide to help you keep track of different Big O complexities.

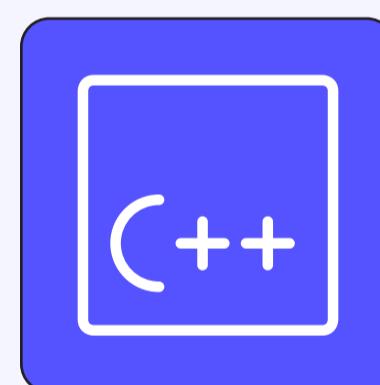
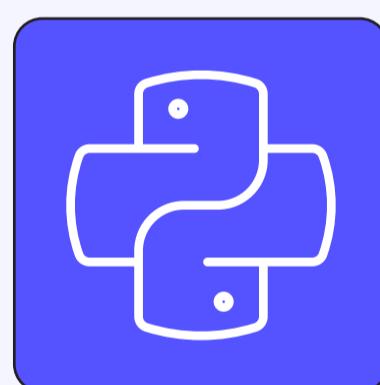
Big O cheat sheet

Algorithmic paradigms, perfected

Study every common algorithm and its use case in our hands-on course: **Mastering Algorithms for Problem Solving**.

Available in [Python](#), [Java](#), and [C++](#).

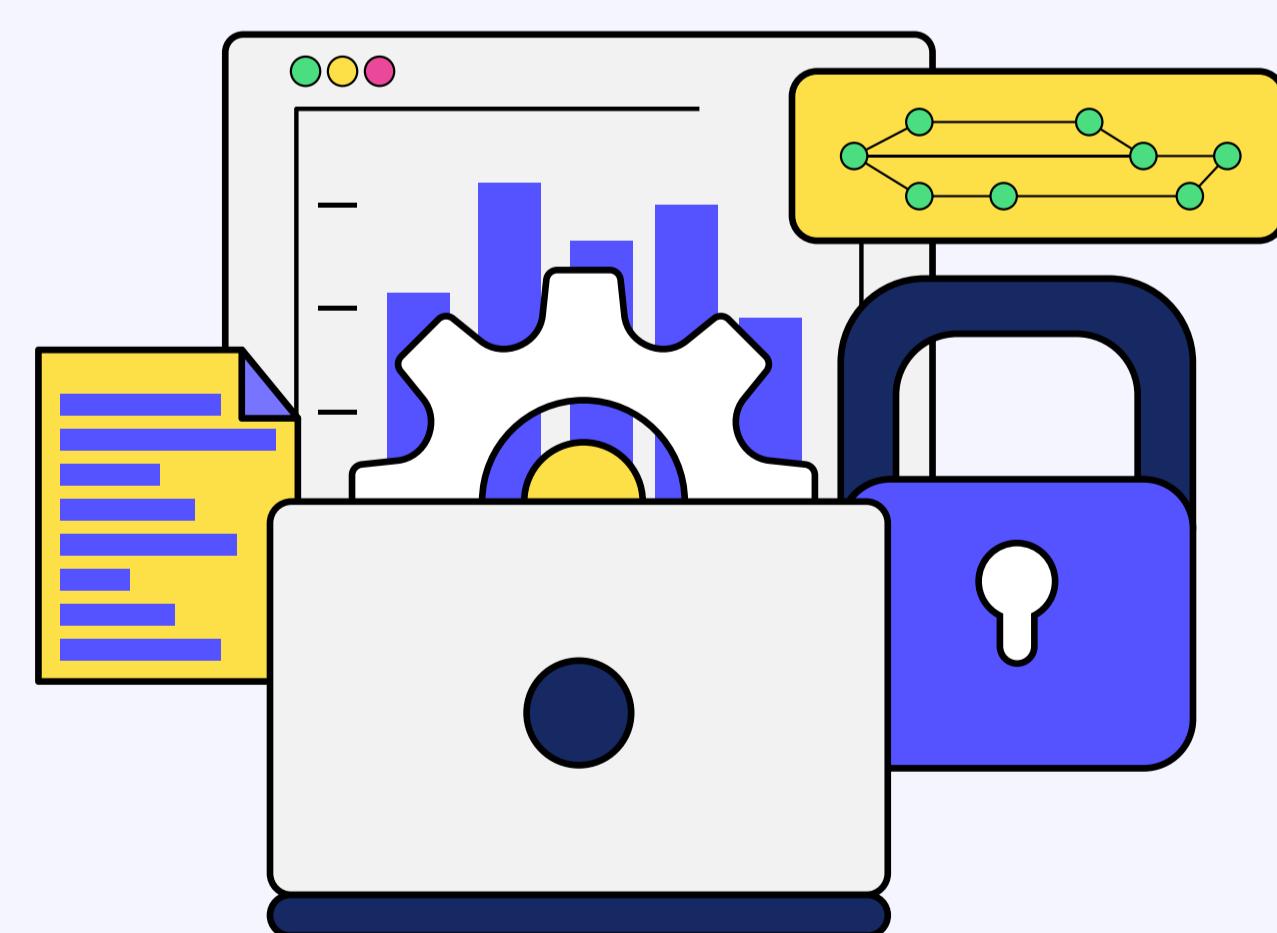
For a primer on data structures, start here:



WEEK 4

Practice with data structures and algorithms

As you're reviewing the basics of data structures and algorithms, start practicing simple problems with the resources listed below. Reviewing the basics will help you internalize these concepts and tackle more difficult problems later.



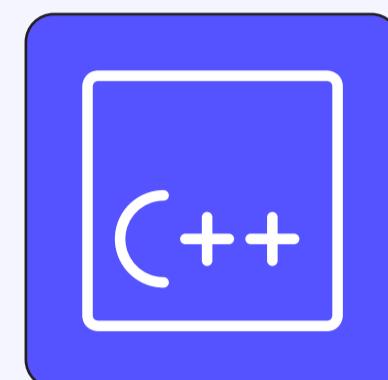
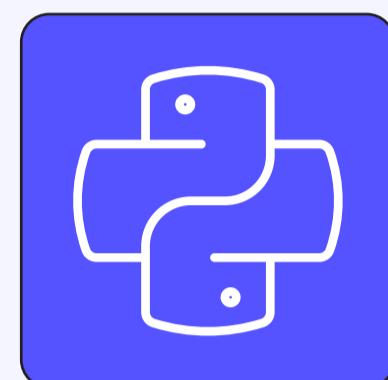
7 Essential Data Structures for Coding Interviews

Master data structures

Review all the common data structures in detail with our hands-on courses, Data Structures for Coding Interviews.

Available in [Python](#), [Java](#), [C++](#), [JavaScript](#), and [C#](#).

To solve problems focusing on algorithms, start here:



WEEKS 5 AND 6

Test your skills with real-world coding interview problems

By this point, you should be breezing through basic practice problems. Now it's time to test your skills by getting hands-on with real-world interview questions.

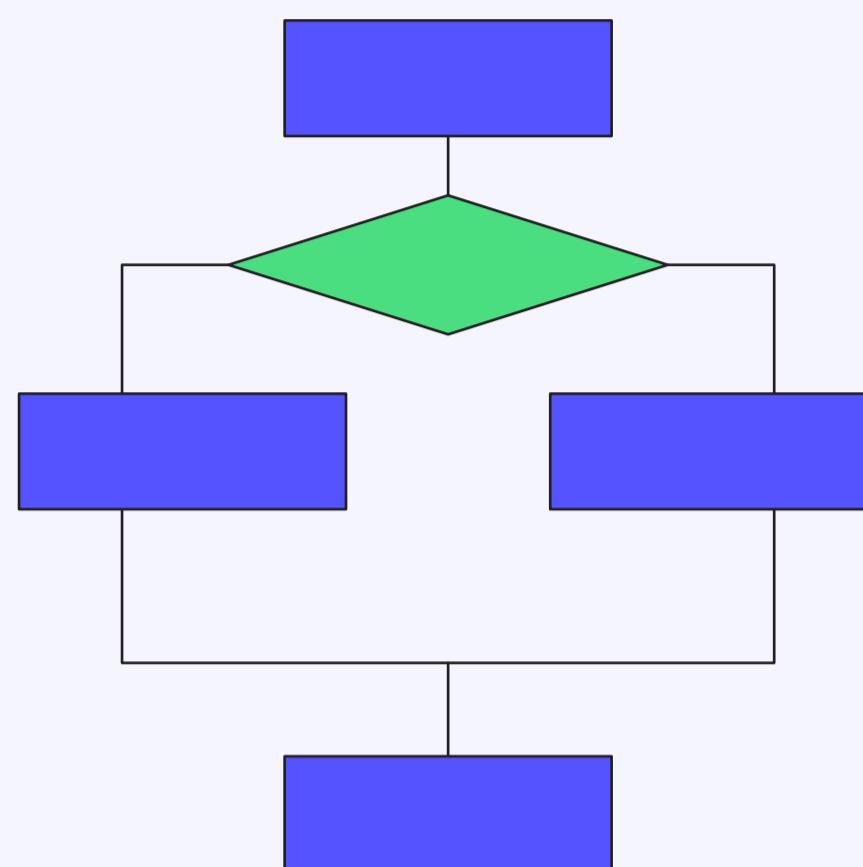


Best practices:

- **Time yourself.** Try to solve your problem in 20 to 30 minutes, but don't be discouraged if some questions take longer at first.
- **Think about the runtime and memory complexities of your solutions.** Your interviewers will likely want you to articulate these complexities and how to optimize them.
- Work on problems using **coding interview patterns**. Almost all questions for a coding interview are built on patterns that serve as a blueprint for solving related problems.

Read about the 7 top patterns here

Depending on your specialization and the company where you're interviewing, the actual questions you get asked may vary greatly. For many general roles, you can expect something related to algorithms and data structures, but it is impossible to say with certainty the actual questions you'll be faced with.



As a result, the most efficient way to organize your prep isn't to complete as many practice problems as possible — it's to master the coding interview patterns behind common questions.

Drilling daily LeetCode problems may help keep your problem-solving skills sharp. However, by internalizing the 26 core patterns that comprise nearly every technical interview question, you will be able to prep more comprehensively, and more efficiently.

14 Must-Know Problems for Coding Interview Prep

Make coding interview patterns a habit

Learn how to solve any possible coding interview problem with our 26 essential patterns. Our popular crash course [Grokking Coding Interview](#) Patterns is available in [Python](#), [JavaScript](#), [Java](#), [C++](#), and [Go](#).

For even faster prep, try our accelerated prep plan: [Educative-99](#) (also available in [Python](#), [JavaScript](#), [Java](#), [C++](#), and [Go](#)).

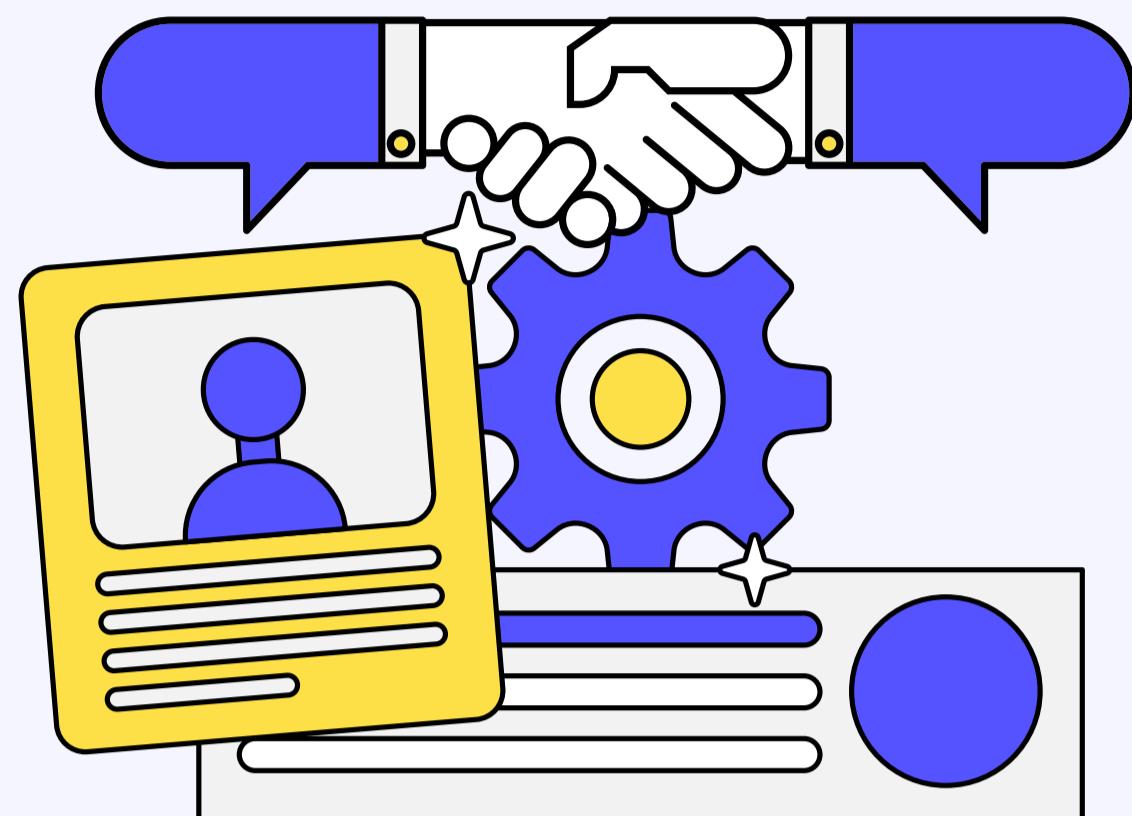
Educative-99 Cheat Sheet

WEEKS 7 AND 8

Object-Oriented Design & Behavioral Interview

The last two weeks of this plan are devoted to studying for the other two rounds of your interview loop:

- Design Interviews
- Behavioral Interviews



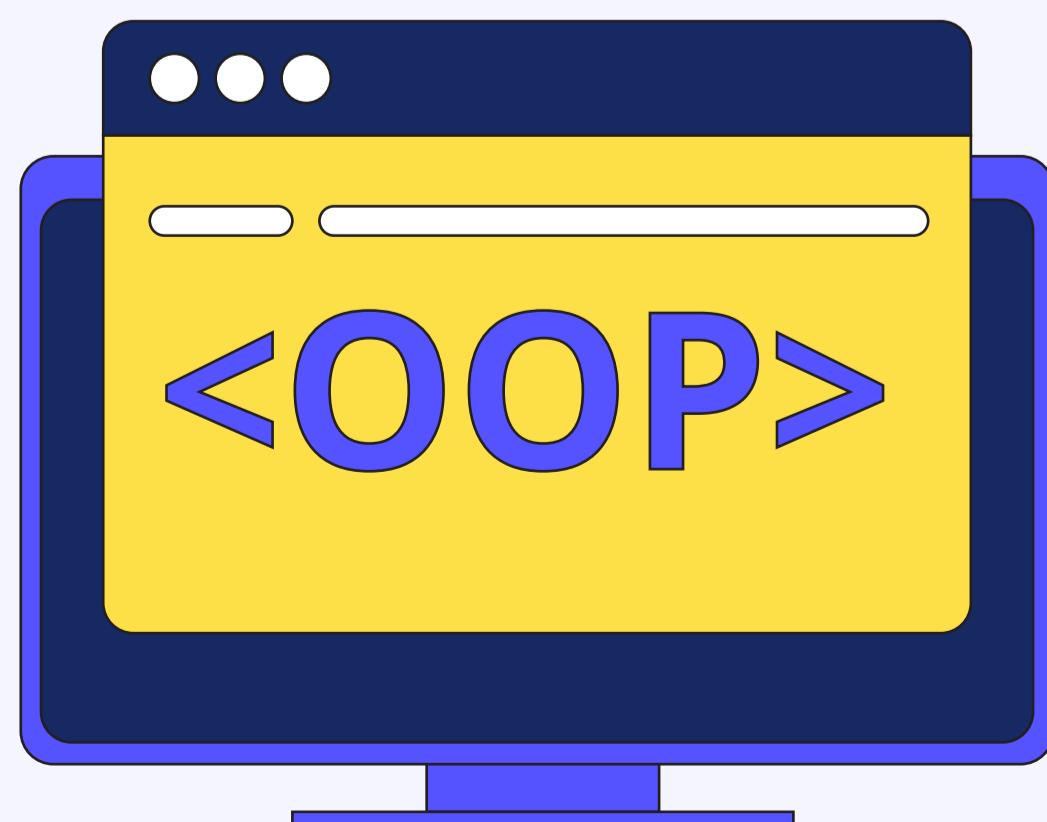
As opposed to coding interviews, which are much more technical in nature, design interviews are all about assessing a **candidate's problem-solving, communication, and collaboration skills**.

Can you navigate tradeoffs, ask clarifying questions, and develop a resourceful solution that satisfies the use case? Can you unpack **functional** and **nonfunctional** requirements? Can you defend the choices you made when pressed?

Behavioral interviews are also designed to assess your soft skills. These interviews often take into account the individual's **company's values and leadership principles**, and are all about determining whether or not you are a good culture fit.

Let's talk about both.

Object-Oriented Design (OOD)



OOD involves implementing specific modules or components and their classes within a more extensive system. We can think of OOD similarly to building the engine of a Formula 1 race car. If the entire car is a distributed system, then the engine's design would be considered a low-level process within that system.

This interview plays an enormous role in determining how strong of a problem-solver you'll be on the job.

The OOD interview will primarily measure two criteria:

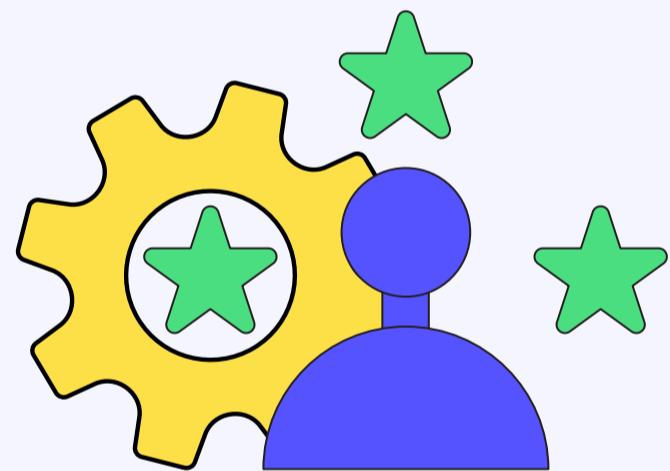
- Can you design a component that successfully interacts with other components in the system?
- Can you use design patterns to create this component efficiently?

Understanding the various design patterns of OOD should be a primary focus of your preparation for any low-level design interview.

Understand real-world systems with OOD

Master design principles and patterns to ace the object-oriented design interview with [Grokkering the Low-Level Design Interview Using OOD Principles](#). Learn a bottom-up approach to break down any design problem using 20+ real-world systems (e.g. Amazon Locker Service, StackOverflow).

Behavioral Interviews



If technical interviews gauge your programming skills, behavioral interviews attempt to discover how you act in employment-related situations or conflicts, both **positive** and **negative**. Behavioral interviews help an employer decide if you're someone they want to work with.

An interviewer may be wondering:

- Is this person calm under pressure?
- Can I rely on this person in a team?
- Will this person treat their peers with respect?

The good news is that just like with technical interviews, the behavioral interview is a skill that can be practiced.

Here are a few tips to help your prep:

1. Familiarize yourself with common behavioral questions

Questions about your prior experience

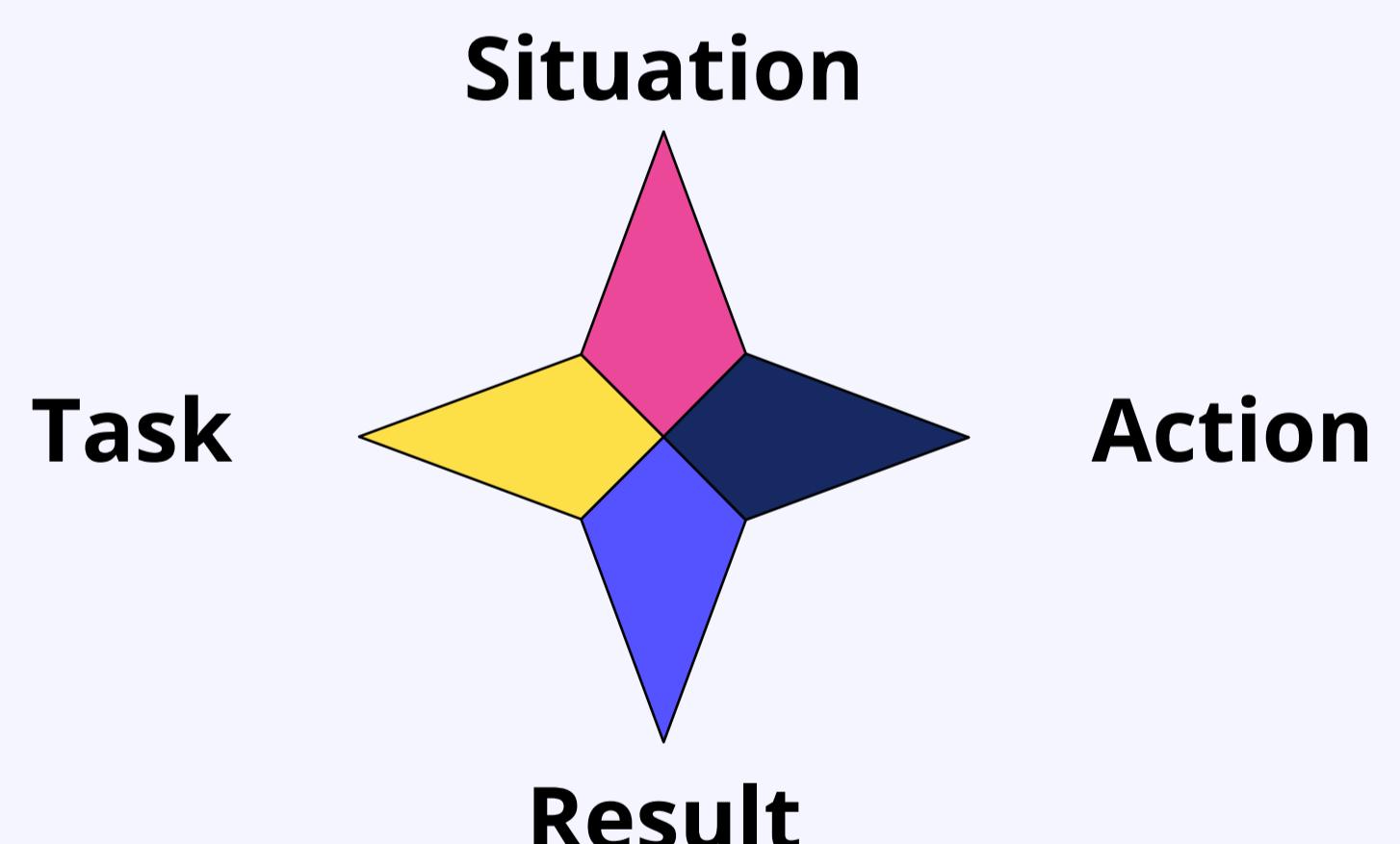
- Hypothetical scenarios (e.g. “what would you do in [blank] situation?”)
- Values-based questions.

2. Research the company ahead of time

Take the time to learn the mission, values, and leadership principles of the company where you are applying (they usually say on their website) CodingInterview.com is a great resource for many top companies.

3. Take the time to practice!

- Practice the STAR method when answering questions (Situation, Task, Action, Result)
- Practice with another person
- Record yourself

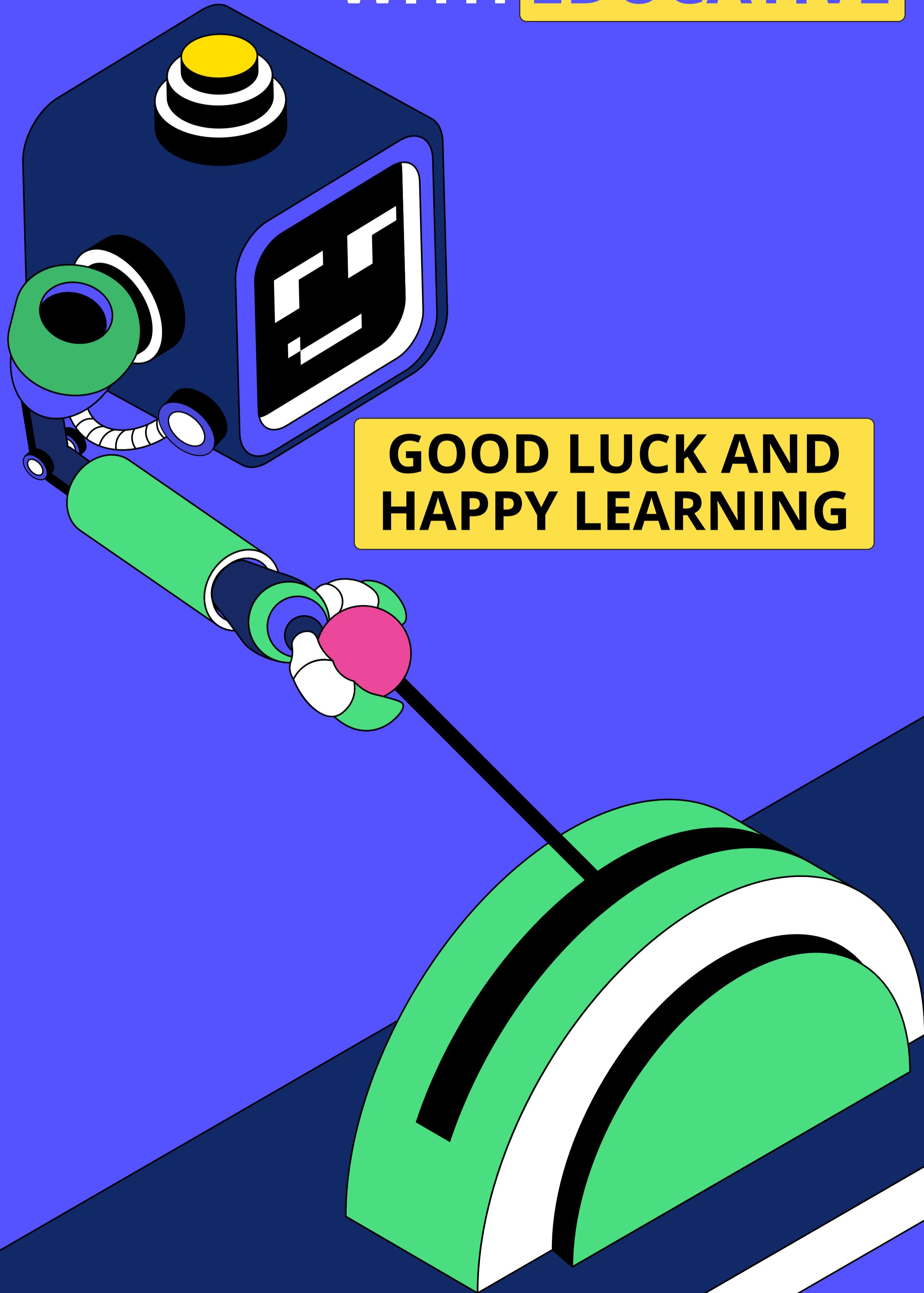


Don't underestimate the power of a strong behavioral interview

Here's your complete guide to behavioral and cultural interviews: [Grokking the Behavioral Interview](https://GrokkingtheBehavioralInterview.com).

It's a **completely free course** that you can use to build your soft skills, just like you would build your technical skills. You can even record a video of yourself to practice your delivery!

JOIN
**2.6 MILLION
DEVELOPERS**
LEVELING UP THEIR SKILLS
WITH **EDUCATIVE**



**GOOD LUCK AND
HAPPY LEARNING**