**Introduction: Interview Skills**

The goal of this unit is to review some key technical interview topics as well as soft skills which employers are looking for in potential candidates. You will practice common JavaScript algorithms and prepare yourself for the whiteboarding interview. You will also learn what soft skills will set you apart and leave a lasting impression on employers. Finally, you will watch several walkthroughs of coding problems often used by tech industry leaders such as Google, Facebook, and Amazon.

After this unit, you will be able to:

* Think on the fly and show your thought processes in solving a problem
* Work your way through several common coding questions
* Respond comfortably to open-ended interview questions
* Set yourself apart and leave an impression on employers

1. **Clarifying the Problem**

Whiteboarding interviews begin with a problem from the interviewer. The interviewee must be confident they understand the dimensions of the problem!

Software development is full of ambiguity. Programming requires concrete deliverables, but company needs can be murky. Even when the need is clear, a feature could have dozens of possible implementations. The ability to clearly define a problem is an important skill to demonstrate.

When the interviewer presents their technical question, repeat the question back to the interviewer in your own words. This gives you a moment to think and will resolve any glaring misunderstandings.

Once you’ve repeated the question, ask every clarifying question that comes to mind.

Assumptions must be communicated to the interviewer so there is agreement on the scope of the problem.

For example, if asked:

**Write a function that returns duplicate characters in string.**

Here are some questions which may come to mind:

* What is the desired return value?
* True|False, a list of characters, or …?
* Do punctuation and spaces count as “characters”?
* Should case be considered?
* are "a" and "A" duplicates?
* Should we be checking for [Unicode](https://en.wikipedia.org/wiki/Unicode) characters?
* Can we assume it’s a 26 character alphabet?

**Instructions**

We’ll apply these steps to a single problem through the rest of the lesson.

Given a list of numbers, return whether the list contains Pythagorean Triplets.

Rephrase this problem in your own words and write that down.

Then, write down every question or assumption you have.

Watch the video to see how we clarified this question.

<https://www.youtube.com/watch?v=xzYgM0eIauA&t=161s>

1. **Producing Inputs and Finding Edge Cases**

When the question is clear, we then produce concrete inputs and outputs. These inputs guide a solution for the remainder of the interview so write them on the board!

You may still be unclear how to solve the problem in code, but it’s certain that given an input, X, your function will produce an output, Y.

Make one input the happy path: input that reflects a common scenario.

For example, you’re asked to write a function which capitalizes the first letter of an input string.

A good input could be "apple", which returns "Apple" because this demonstrates the function’s purpose.

If the input were "Apple", it would return "Apple". That’s correct but less informative.

Also think about edge cases, or inputs which do not reflect a common scenario and may cause problems.

For the capitalization function, what should you return if given None as input? Or a number?

Instructions

Write inputs and outputs for the Pythagorean Triplet problem.

Try to give yourself a few different cases.

Watch the video to see how we made inputs/outputs.

<https://www.youtube.com/watch?v=tccfJmGM0XI&t=25s>

1. **Writing the Outline**

It’s time to start breaking down the problem by category.

Given a question which requires the use of a stack, what do you know about stacks? Have you encountered other problems that use stacks and how were they solved?

Is this a searching question? Can you sort the input and will that help? Does this problem sound like it can be modeled as a graph, with vertices and connected edges?

Understanding the applications of different data structures is very useful! The more questions you practice, the more you will be able to see patterns between problems.

This step varies the most because it requires details of the specific problem, but regardless of the question make certain you are communicating with the interviewer as a potential co-worker. **Show them your thought process!**

During this step, the interviewer may make suggestions on how to proceed. Acknowledge the interviewer and incorporate their suggestions into your approach.

**Don’t disregard their input! It gives the impression you would be difficult to work with on the job.**

When you and the interviewer are satisfied with a workable solution, write the steps next to your input. Follow these steps as you write code on the board.

Instructions

Come up with an outline of how to solve the Pythagorean Triplet problem.

**Don’t worry about efficiency, just aim for high-level steps which will produce the correct output.**

Watch the video when you’re ready to move on.

<https://www.youtube.com/watch?v=yt-YB_9ZHUE&feature=emb_title>

1. **Coding the Solution**

We’ve finally come to the portion where you’ll write some code! Previous steps improve your odds for success because you have an outline and clear inputs for testing.

Writing code on the board is a collaborative process. Refer to your outline and explain the step you’re implementing.

The goal is to be facing the interviewer when talking through the implementation and facing the board when you’re writing the code.

Try to avoid writing code in silence or narrating at a low level like “for… i… in… range… length of the list… colon” when writing for i in range(len(input\_list)):.

When you’re finished with the implementation, look it over for any mistaken syntax or logical errors.

Instructions

Code the solution to the Pythagorean Triplets problem by writing it out on paper. Use your outline from the previous exercise as a guide.

Before writing each step on paper, say what you will do out loud. Better to practice by yourself than in a real interview!

<https://www.youtube.com/watch?v=hwoGhge9y5I>

1. **Testing with Inputs**

With a working implementation on the whiteboard, use the test inputs to walk through the evaluation of your code.

Write out any temporary variables on the board and update them when they change during execution.

This is another opportunity to showcase your communication skills, and will help you catch any logical errors which you may not have noticed in earlier steps.

**If you catch an error, don’t panic! Mistakes happen**. Explain the issue and talk through what you can do to fix the bug.

Your interviewer may want to see you write the correction or they may be satisfied with the explanation.

Instructions

Use your test inputs and implementation to **talk through your solution**.

Again, **say each line aloud** so you become more comfortable speaking about the code you’ve written.

Track the variables used and update them as you go.

<https://www.youtube.com/watch?v=qZpIz_lmp4Q>

1. **Analyzing Time and Space Complexity**

You’re satisfied with your implementation and you’ve demonstrated how it works, but you’re not quite done.

Analyze the time and space complexity of the solution. With this step you are demonstrating that you care about the efficiency of your code.

Explain your code’s big O notation. If you can optimize to a more efficient runtime, explain how that would work. If you can’t optimize, explain why it’s not possible.

Instructions

**Talk through the big O time and space complexity of your solution.**

Explain how an optimization would be possible or not.

<https://www.youtube.com/watch?v=bn5ORboA-6Y>

**Review**

Technical interviews judge the candidate’s ability to perform technical aspects of the job. Whiteboarding is a type of technical interview which requires the candidate to code a solution without the assistance of a text editor.

Coding an efficient solution is important, but the ultimate goal as an interviewee is to convince the interviewer you would be a desirable co-worker.

A desirable co-worker is able to articulate their thought process and collaborate without friction.

Our whiteboarding strategies revolve around communication at every step.

The steps are:

1. **Clarify** the Problem
2. Create **Inputs**
3. **Outline** the Solution
4. **Code** the Solution
5. **Test** the Solution
6. **Analyze** the Solution

Instructions

Practice makes perfect!

Run through these steps with a new practice problem:

Write a function which prints every number from 0 up to the given input. If divisible by 3, print "Fizz" instead of the number. If divisible by 5, print "Buzz". If input is divisible by 3 AND 5, print "FizzBuzz".

<https://www.youtube.com/watch?v=eADUTW9mrEo>

Quiz: Technical Interviews: Whiteboarding

**What is the next step after clarifying the technical question asked by the interviewer?**

* Create example inputs and outputs to the function you will write; Sample inputs help guide a solution by defining exactly what the function will do.

**When considering inputs for your function, what are edge cases?**

* Inputs which could cause issues with the implementation of your code; An edge case can be an invalid input, or inputs which cause your code to execute in a way you had not expected.

**A technical interview evaluates a candidate’s \_\_\_?**

* Ability to write code; Technical interviews focus on the programming responsibilities of the position.

**What is the first thing you should do when an interviewer asks you a technical question?**

* Repeat the question back to the interviewer in your own words; This ensures you’ll both be in agreement on the problem to be solved.

**Code Review: Interview Skills**

<https://discuss.codecademy.com/t/article-giving-and-receiving-code-review/521901/4>

[Sieve of Eratosthenes](https://www.codecademy.com/paths/back-end-engineer-career-path/tracks/becp-22-interview-skills/modules/wdcp-22-javascript-algorithm-practice/articles/sieve-of-eratosthenes-javascript)

* **How will I remotely give a code review?**

There are lots of tools available to help make the code review process easy. [Here’s a list of 12 code review tools.](https://kinsta.com/blog/code-review-tools/) Alternatively, you can post your code in a direct message in either [the forums](http://discuss.codecademy.com/) or Discord, and provide one another feedback right then and there.

* **What will we work on?**

We’ve made a few suggestions that align with where you are in the path. However, feel free to choose a different project in the Path or go over a personal project instead.

* **Do I have to do this?**

Nope! But the curriculum developers at Codecademy believe this is a strong pedagogical tool for cementing your learnings as a new developer. Additionally, code reviews are an important part of the software development process when working on a professional team.

[Cheatsheet](https://www.codecademy.com/learn/paths/back-end-engineer-career-path/tracks/becp-22-interview-skills/modules/wdcp-22-technical-interviews-whiteboarding/cheatsheet)