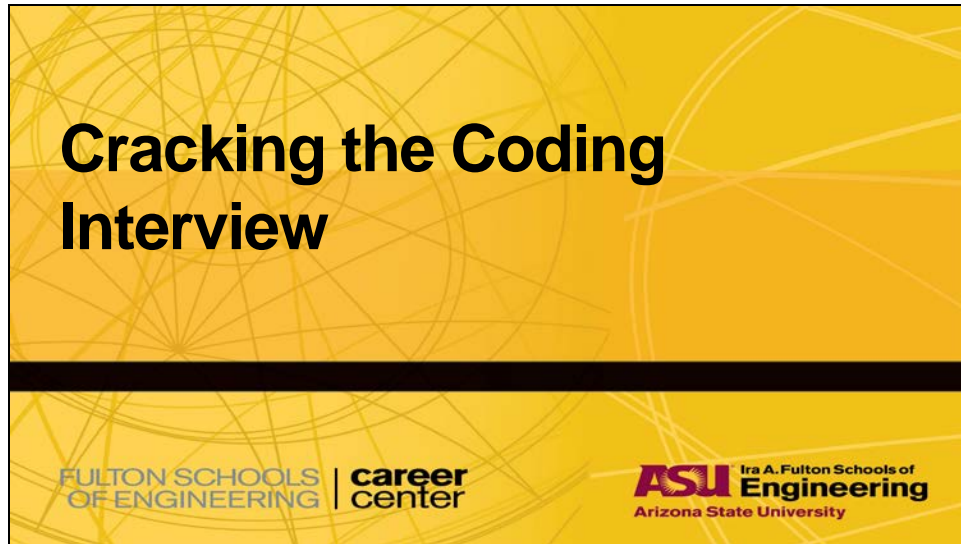


Slide 1



This presentation is for technical majors seeking technical degrees.  
Other fields? Hiring practices differ, so the content and format may differ from the advice in this presentation.

## **Your Goal, at any interview**

- **Confident**
- **Competent**
- **Enthusiastic**
- **Someone that I/we can like and trust**

These are the goals for any interview.

Coding is a special form of interviewing, and has certain standards, expectations, and protocol.

Note that confidence is important: confident people find ways to complete work and solve problems, even when they lack competence. They find ways to close gaps by using resources, such as peers and other sources of information. Applicants who do not appear confident raise doubts about their ability to succeed in the workplace.

Competence: is judged on resume items, and is the primary focus of the coding interview.

Note, the competence for coding interviews is as much about SOLVING PROBLEMS as it is CODE ABILITY.

Enthusiasm: are you interested? I don't want to go through the work of offering you the job if you are not interested. How do you show that? By indicating that you have taken some time to investigate the organization, the industry, ... and that you have determined how your experience/expertise can fit into what work we have / problems we solve / .

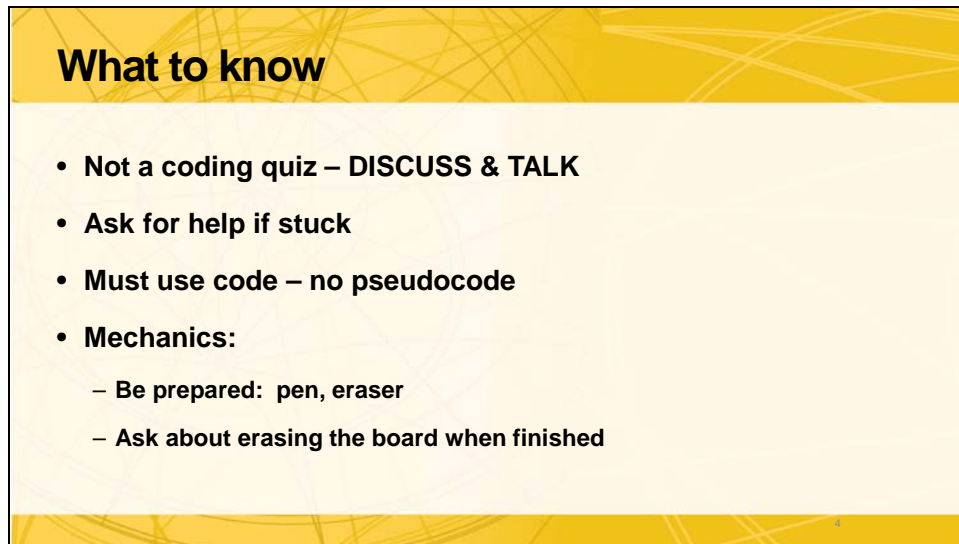
Like and trust? Who wants to work with disagreeable co-workers? Not me either. Trust? Untrustworthy co-workers create work for managers and everyone else. This is the reason that errors or discrepancies on your resume will usually disqualify you for a position.



## **Outline**

- 1) What to know, what to expect, and how to react**
- 2) Sample coding interview questions to work**
- 3) References and resources**

For the coding interview, we are going to cover these items: basic what-and-how-to items, a sample of coding exercises, and how to learn more.



## What to know

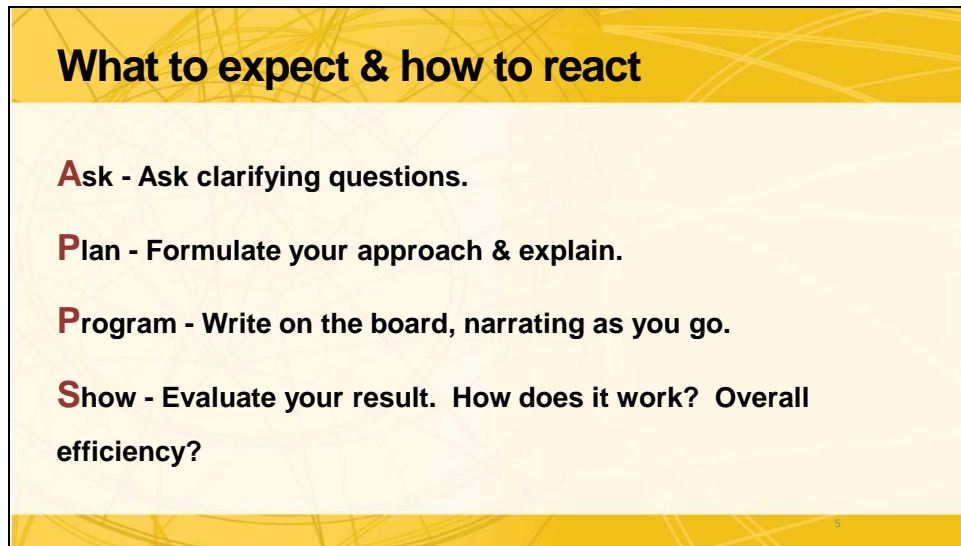
- **Not a coding quiz – DISCUSS & TALK**
- **Ask for help if stuck**
- **Must use code – no pseudocode**
- **Mechanics:**
  - **Be prepared: pen, eraser**
  - **Ask about erasing the board when finished**

How does this work? You will be on line (using an on line document that you share with the interviewer) or in an interview room with an interviewer (using a laptop or a whiteboard). They will give you a problem to solve.

- You do not jump up, go to the board, and start coding. No reward for fastest answer. This is a problem solving exercise. This is a conversation, like one you would have with a colleague about a technical problem in a project.
- Ask for help if you get stuck. The interviewer can give you hints. Again, this is a conversation.
- Code only. There may be a preferred language, or you may get to pick.
- Mechanics: you want to be confident. For a white board, make sure you are equipped with a pen and something to erase the board, and move the chairs/stuff out of your way if needed.
- At the end, ask if you should erase the board. Don't assume you do, and don't assume you don't. They might want to reference it later, or they might need the space immediately.

Why are all these details important? Remember: confident – competent - ...? It's hard to be confident if you do not have all the mechanics of the interview in your favor. Board marker is dry or yucky color or --? This will take time out of your interview to resolve. Wiped the board with your hand? Oops, got marker dust all over your hand? Not the way to be confident.





## What to expect & how to react

- A**sk - Ask clarifying questions.
- P**lan - Formulate your approach & explain.
- P**rogram - Write on the board, narrating as you go.
- S**how - Evaluate your result. How does it work? Overall efficiency?

Here's what you do: Build your "APPS"!

We are going to cover 4 points on the basics, and then we will use examples to illustrate.

**Ask:** Ask questions about the issue. This is a problem-solving exercise. You must be sure that you understand the problem and any boundary conditions, limitations, and requirements

**Plan:** Explain your plan of attack. You want to solve the problem the simplest way. Brute force is ok.

**Program:** State all assumptions as you go and describe what you are doing.

**Show:** Discuss what you might do for a better result, if you have another option.

And be prepared for the interviewer to ask you some questions that make the problem different. For example, if the data set was 10x larger: how would the solution you proposed (when the assumption for the data was smaller) work?

**Note:** you are not assigned extra credit for solving the problem faster. Speed is not the criteria for this exercise. Take your time, be thorough, and be clear with the communication of the problem and the solution you have.

**Samples**

- **Example question: what would you do to respond?**

**1. Develop a program to test if two words are anagrams.**

Student should ask about desired arguments and output (input strings, output Boolean).

Student should ask about case, numbers, whitespace, etc.

**Solutions:**

**Easy solution:** Sort both strings and do a character by character comparison. Depending on the sorting algorithm, the best time complexity through this method is  $O(n \log n)$

**Intermediate solution:** Map each of the 26 English characters to a unique prime number. Then calculate the product of the string. By the fundamental theorem of arithmetic, 2 strings are anagrams if and only if their products are the same.  $O(n)$

**2. Given a linked list of unknown length, write an algorithm that returns the n to last element.**

Student should ask about desired arguments and output (input linked list and integer, output linked list node)

**Solutions:**

Easy solution: Iterate over linked list to find the length. Perform a second iteration ( $\text{length} - n$ ) and return the node.

Traverses the linked list twice

Intermediate solution: Use two iterators (fast and slow). Iterate over the linked list and only increment the fast iterator till it is on node n. Then, start iterating over both iterators till the fast iterator gets to the last node. The slow iterator should be at node



(length – n)

Traverses the linked list only once

- **Given an array of integers, write a function to remove duplicates.**

Note the problem statement: *get rid of the duplicates.*


- Student should ask if the original order of the array must be preserved.
- Student should ask if the duplicates themselves have any meaning – do they need to be evaluated (so capture them somewhere) or just discarded?

If the order does not need to be preserved and the duplicates do not need to be captured, this is a simple exercise: sort the list and then test elements against adjacent elements.

If otherwise, solution will require more complex comparison of two arrays.

## Recommended References

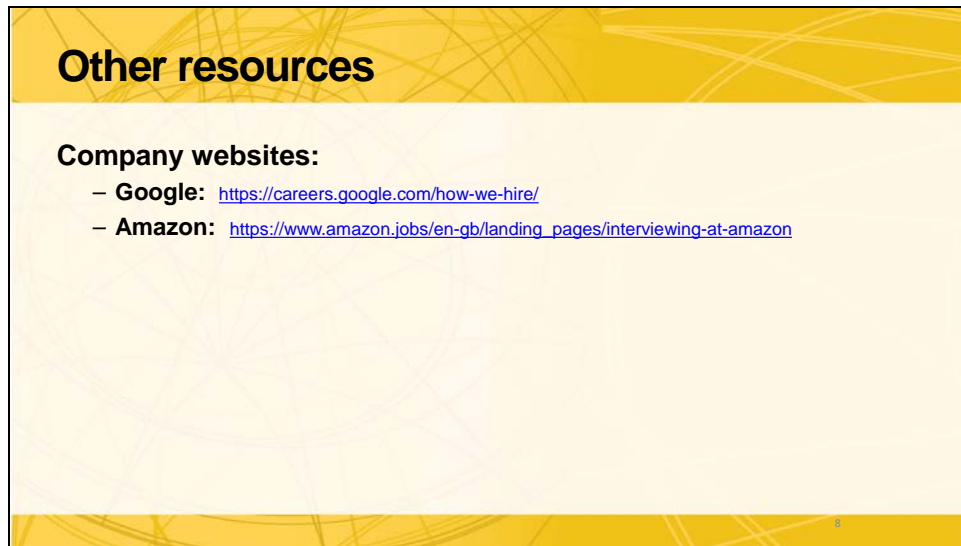
- **What can I read?**
  - Available at Noble Library as reference
- **Where do I find examples?**
  - Codecademy
  - LeetCode
  - HackerRank



“Cracking the Coding Interview” can also be found on line (PDF version). Tell students to read the first part of the book, not just the problems ;). The material in the first part of the book contains excellent advice for the broad subject of “interviewing” in general.

<https://epiportal.com/Ebooks/Cracking%20the%20Coding%20Interview%2C%204%20Edition%20-%202015%20Programming%20Interview%20Questions%20and%20Solutions.pdf>

Websites: Direct students to Codecademy, LeetCode, HackerRank, and any others you might be aware of.



## Other resources

**Company websites:**

- **Google:** <https://careers.google.com/how-we-hire/>
- **Amazon:** [https://www.amazon.jobs/en-gb/landing\\_pages/interviewing-at-amazon](https://www.amazon.jobs/en-gb/landing_pages/interviewing-at-amazon)

Remind students to always review information on the company website. Probably a good idea to review sites of other companies as well.

## Additional Information

- **Handshake** <https://asu.joinhandshake.com/login>
  - Learn about career fairs, company information sessions and other career events
  - Apply to internships and jobs
  - Schedule an appointment with the career center
- **Fulton Schools Career Center website** <https://career.engineering.asu.edu/>
  - 24/7 access to presentations and tools
- **Optimal Résumé** <https://career.engineering.asu.edu/optimal-resume/>
  - Submit your résumé for review without an appointment
- **CareerSpot videos** <https://career.engineering.asu.edu/careerspots/>
  - Short videos from recruiters and career services for quick career topics
- **GoinGlobal** <https://career.engineering.asu.edu/goinglobal/>
  - Worldwide opportunities, visa petition history from US employers
- **InnerCircle** <https://innercircle.engineering.asu.edu/category/career/>
  - Weekly enewsletter about all things Fulton, including timely career topics and events
- **Fulton Career Center on Facebook** <https://www.facebook.com/fultoncareercenter/>
  - Featured events and opportunities

Here are some notes