CoGrammar

Welcome to this session:

Coding Interview Workshop - Problem-Solving Approaches

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Coding Interview Workshop

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Coding Interview Workshop

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: **Feedback on Lectures**
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- Develop a **structured approach to solving coding problems** by breaking them into clearly defined steps.
- Apply techniques such as **input/output analysis**, **constraints evaluation**, and **edge case identification** to fully understand a problem before coding.
- Use **pseudocode**, **flowcharts**, and **dry runs** to plan solutions systematically.
- Learn to identify when **brute force solutions** are acceptable and when **optimization** is necessary.
- **Optimize solutions** using more efficient data structures or algorithmic paradigms.
- Practice **iterating on solutions based on feedback**, debugging effectively, and recognizing common pitfalls in coding interviews.

Lecture Overview

CoGrammar

What is your usual first step when solving a coding problem?

A. Start coding immediately

B. Read the problem multiple times

C. Write down inputs and expected outputs

D. Sketch a flowchart or pseudocode

CoGrammar

How often do you consider edge cases before writing code?

A. Always

B. Sometimes

C. Rarely

D. Never

CoGrammar

Problem Solving Under Pressure!

Imagine you're in a coding interview, and the interviewer asks you to **solve a problem you've never seen before**. You are given a white board and are asked to **walk them through your thinking** while you are solving the problem. You have **10 minutes** to solve the problem and present your solution.

- How do you approach it?
- Do you start coding right away?
- Do you take a step back and break it down?

CoGrammar

Problem Solving Under Pressure!

In the beginning weeks of the Coding Interview Workshops, we used the abbreviation "**APC**" to help us tackle problems:

1. Note your **Assumptions**
2. **Plan** your solution
3. Write the **Code**

This method allowed us to **take control** of the problem-solving process by slowing down and understanding the problem **before** looking at a solution. This **prevents errors** and stops us from **wasting time**. In today's lecture, we'll refine that method.

Key Techniques

- **Input/output analysis** - Understanding constraints and expected outputs.
- **Edge case identification** - Handling edge cases like empty inputs or large datasets.
- **Brute force vs. optimization** - Knowing when a naive approach is acceptable and when optimization is necessary.
- **Debugging & Iteration** - Testing, logging outputs, and refining solutions.

CoGrammar

Steps to Success

1. **Understand the Problem**
   - Read the problem carefully.
   - Identify inputs, outputs, constraints, and edge cases.
   - Ask clarifying questions if needed.
2. **Plan Before You Code**
   - Use pseudocode or flowcharts to outline the solution.
   - Determine if the problem allows for brute-force solutions or needs optimization.

CoGrammar

Steps to Success

3. **Implement a Solution**
   - Code step-by-step, following the plan.
   - Test using example cases.
4. **Optimise**
   - Identify inefficiencies.
   - Use better data structures or algorithmic paradigms to improve performance.

CoGrammar

Practice the Plan

Let's practice our method by solving the following problem:

Given an array of integers, find two numbers that add up to a target sum.

Then we'll do the following problems together:

- Median of two sorted arrays
- 3Sum
- Valid Parentheses

CoGrammar

Do you feel more confident in structuring your approach to coding problems?

A. Yes

B. Somewhat

C. Not really

D. No

CoGrammar

What was the most valuable part of today's lecture?

A. Learning a step-by-step problem-solving approach
B. Understanding brute force vs. optimized solutions
C. Seeing a real coding problem solved
D. Practicing with LeetCode-style problems

CoGrammar

Homework

Practise the skills we've developed by completing the rest of the LeetCode questions:

- Practise speaking through your solutions and explaining how you approached each problem.
- In the next lecture we'll be covering the topic: "Algorithmic Paradigms"
- You can have a look at the following LeetCode questions to prepare:
  - Example 1
  - Example 2
  - Example 3

Summary

Problem Solving Steps:

1. Understand the problem
2. Plan your solution
3. Implement your solution
4. Optimise!

Tips to remember:

- ★ Always understand the problem before coding.
- ★ Use pseudocode, flowcharts, and dry runs for planning.
- ★ Brute force is a starting point, but optimization is key.
- ★ Efficient data structures like hash tables can significantly improve performance.
- ★ Iteration and debugging are critical for refining solutions.

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the Graphs, should you have any.

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education