Technical Interview Preparation in Summer (TIPS)

Lecture 4: Interview Techniques, Part 2 (Coding)

Congratulations on completing your first set of mock interviews! 🎉🎊🥳

- You probably felt like it was difficult, awkward, and uncomfortable.

  That's normal! And because you're just starting out.
- Treat technical interview preparation as *training a muscle*
  - Feels horrible when you first start exercising it
  - But as you do it more and more, you start to get more comfortable with it and more confident in your own skills
- Key thing is: It's difficult, but you will be glad you got started with it.

Outline for Tonight

- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

Outline for Tonight

- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

Admin (Writing Emails)

- Write emails to soc-tips@googlegroups.com and not to individual team members, though you may address us individually when writing in

Admin (Mock Interviews)

- From now onwards, you will have to do a mock interview **every week**
- Reach out to your partner **early**
- Please write to us if your mock interview partner is unresponsive and we’ll find another partner for you
    - Unresponsive meaning they don’t reply after 2-3 days (give them time to do so)

Admin (Advance Notice)

- If you are going to be **unavailable** during any of the upcoming weeks (e.g. going overseas), please let us know **early**
- Write in **before** the week starts and we group you for activities
- It is very **unfair** to your partner(s) if you only write in about your unavailability **after** being grouped together; also incurs a lot of **additional workload** for the teaching staff

Admin (Homework Announcements)

- Homework announcements are posted every Monday
- Please read them **carefully** and **ask** us if you’re unsure about anything

Outline for Tonight

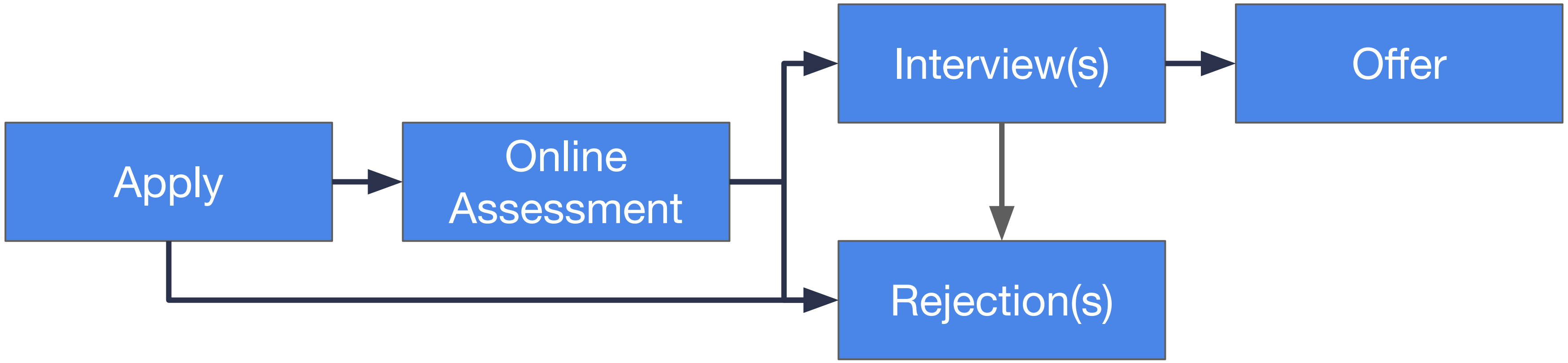
- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

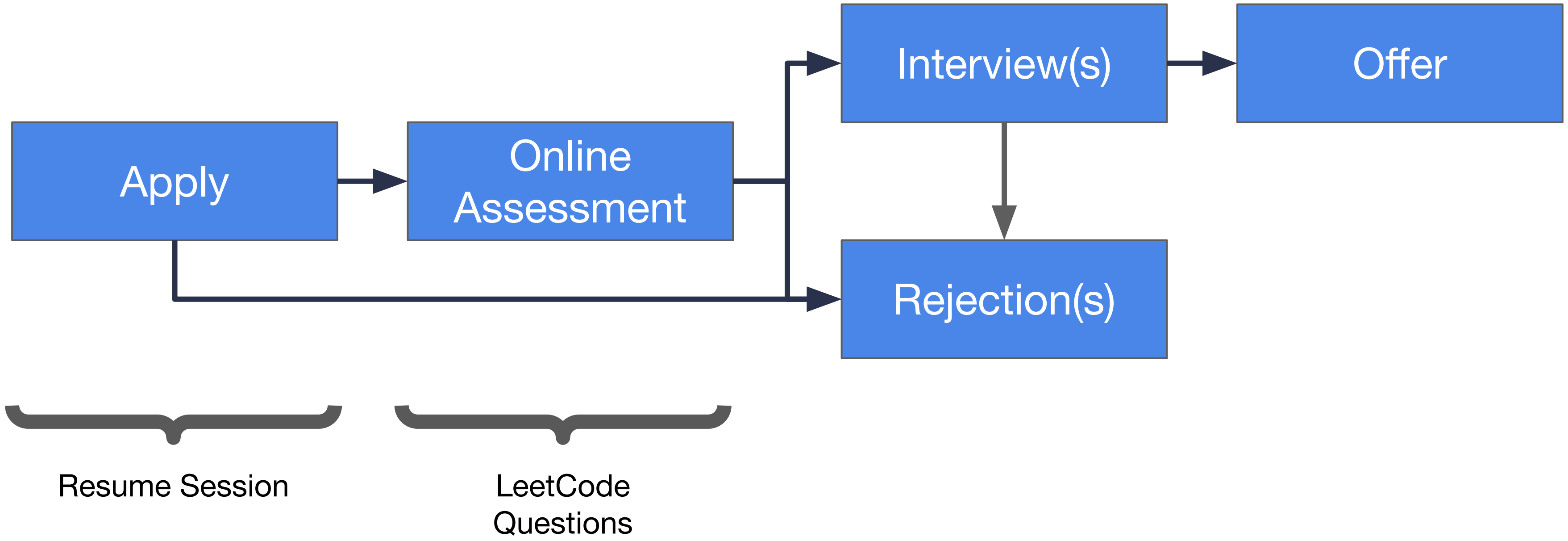
DISCLAIMER

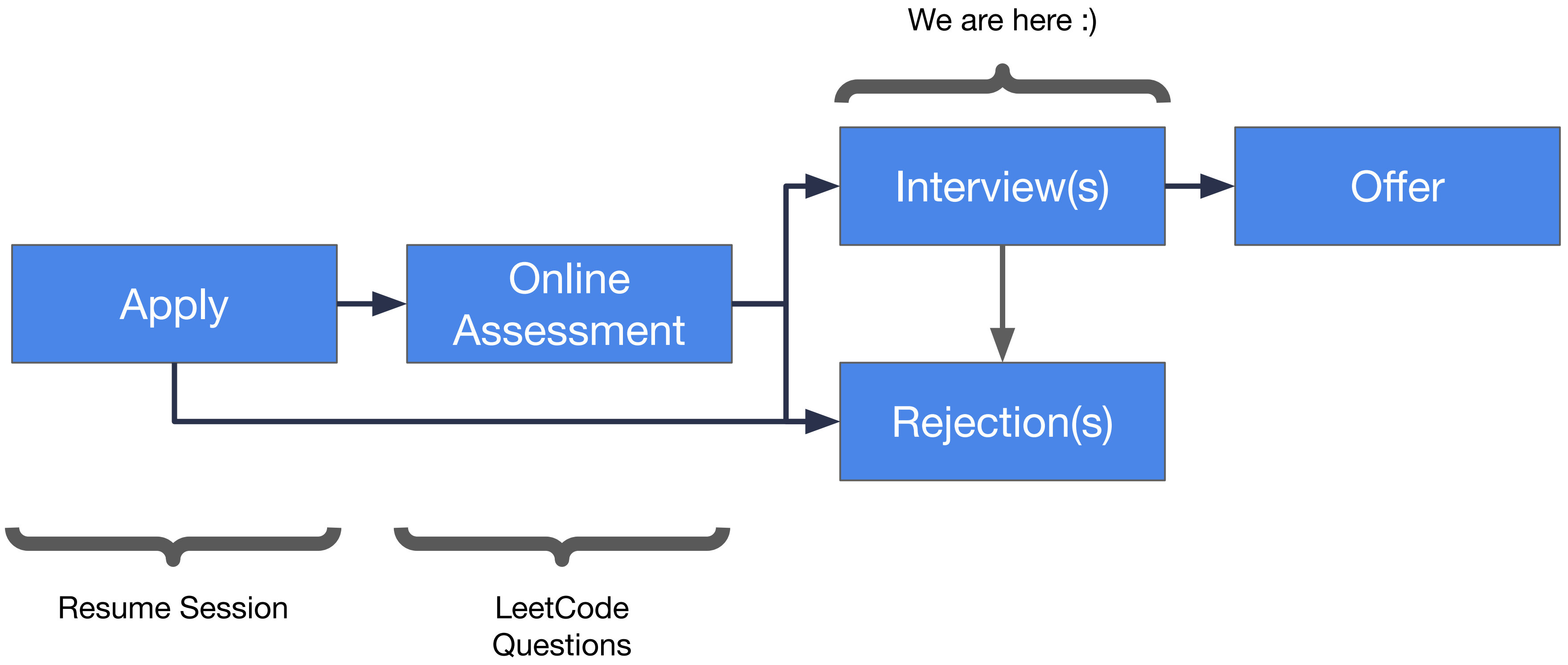
By no means is TIPS an “absolute guide” on how things will be!

Every company is different, and every interview is different.

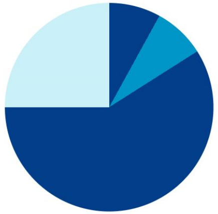
BUT many of the things that we share will be quite **transferrable**.


Apply
Online Assessment
Interview(s)
Offer
Rejection(s)


Apply
Online Assessment
Interview(s)
Offer
Rejection(s)
Resume Session
LeetCode Questions


We are here :)
Apply
Online Assessment
Interview(s)
Offer
Rejection(s)
Resume Session
LeetCode Questions

Overview of Techniques

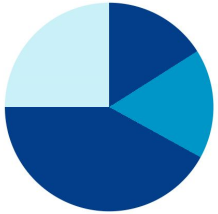


The first 5…

Proactive Communication

Designing Your Algorithm

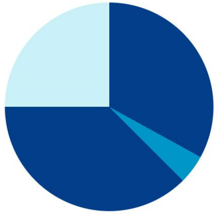
Last week



The next 10…

Maximising Collaborative Editors

Talking As You're Coding

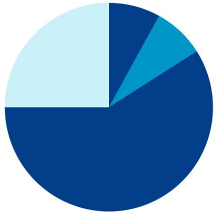


The last 2-3…

Handling Mistakes

Testing Your Code

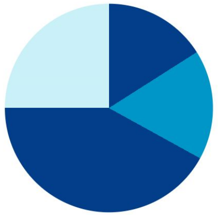
Overview of Techniques



The first 5…

Proactive Communication

Designing Your Algorithm

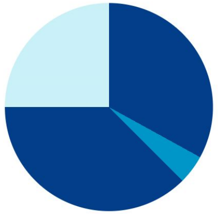


The next 10…

Maximising Collaborative Editors

Talking As You're Coding

This week



The last 2-3…

Handling Mistakes

Testing Your Code

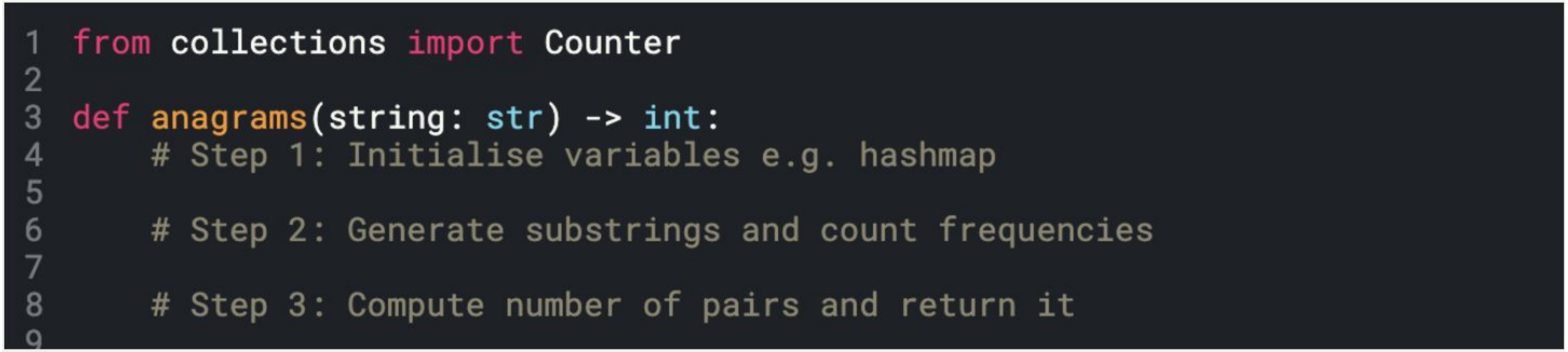
Overview

Everything you write on collaborative editors will **remain in view** for the interviewer :O

Leave **comments/workings** that allow interviewers to know what you are doing at any point in time.

Quickly type out **diagrams** and **examples** (that are not too complex) that both you and the interviewer can easily interact with.

Scaffolding



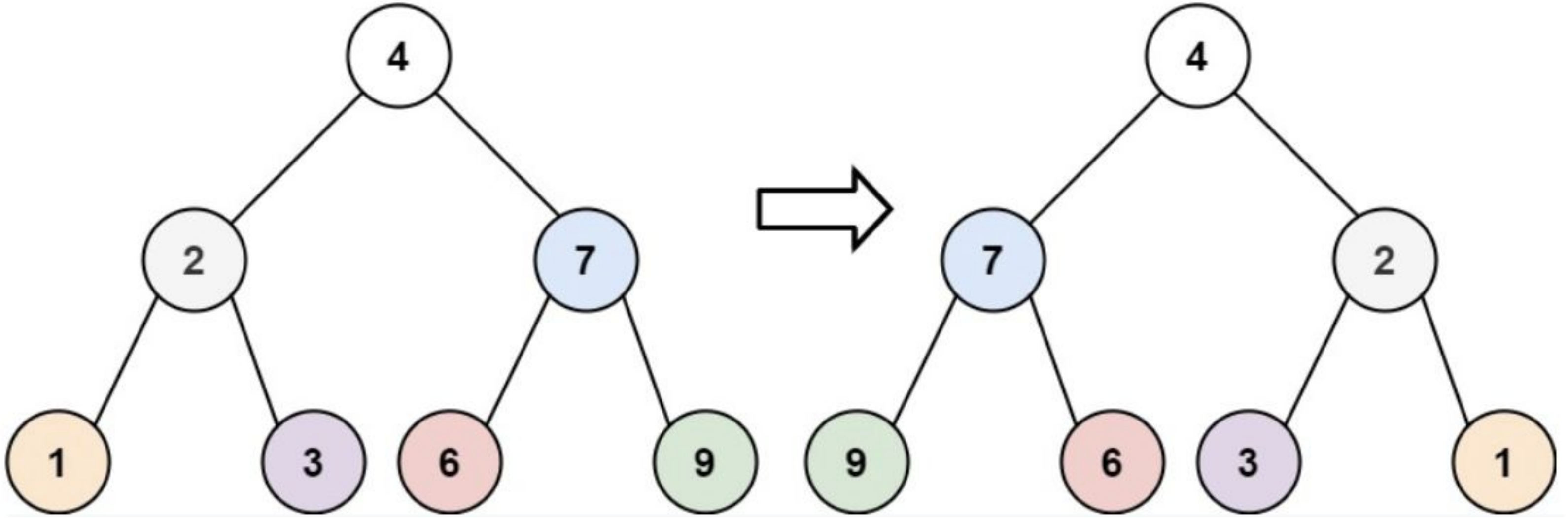
```python
from collections import Counter

def anagrams(string: str) -> int:
    # Step 1: Initialise variables e.g. hashmap

    # Step 2: Generate substrings and count frequencies

    # Step 3: Compute number of pairs and return it

```

Let interviewers know what you're going to do!

This part can ideally be copied from your algorithm brainstorming earlier.

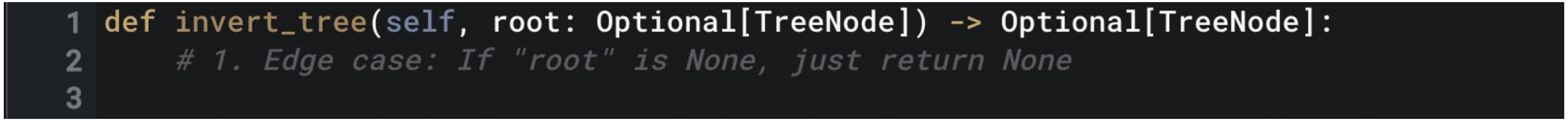
Scaffolding Example

Given the root of a binary tree, invert the tree, and return its root.



Scaffolding Example

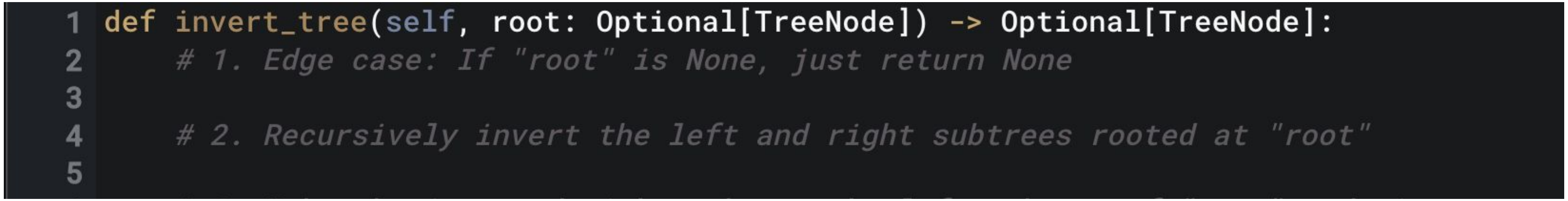
Given the root of a binary tree, invert the tree, and return its root.



```python
def invert_tree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
    # 1. Edge case: If "root" is None, just return None

```

Scaffolding Example

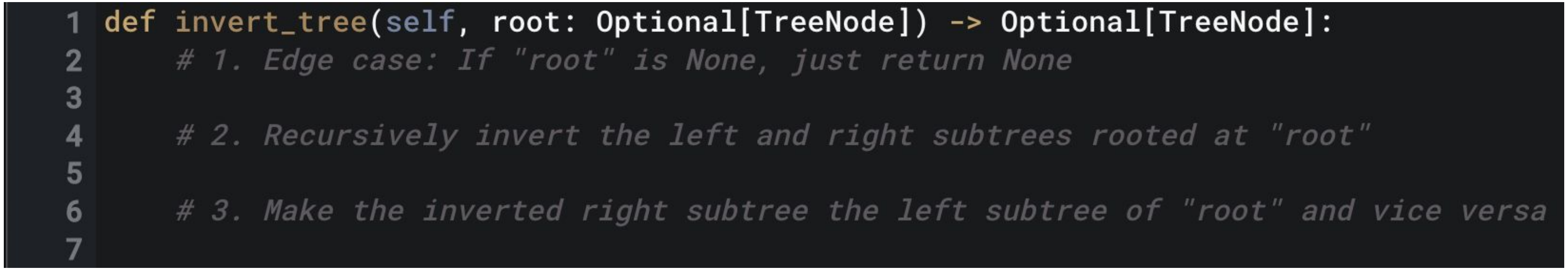
Given the root of a binary tree, invert the tree, and return its root.



```python
def invert_tree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
    # 1. Edge case: If "root" is None, just return None

    # 2. Recursively invert the left and right subtrees rooted at "root"

```

Scaffolding Example

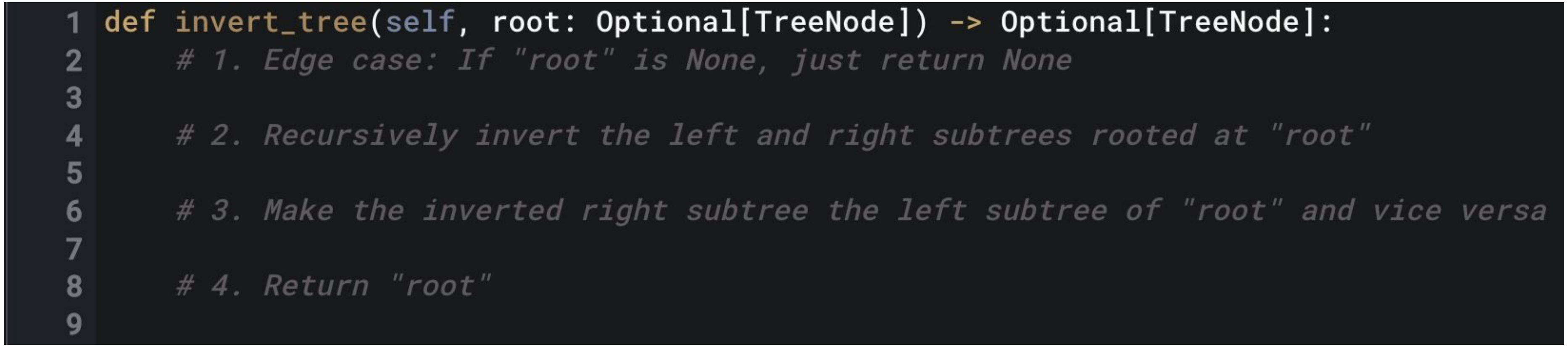
Given the root of a binary tree, invert the tree, and return its root.



```python
def invert_tree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
    # 1. Edge case: If "root" is None, just return None

    # 2. Recursively invert the left and right subtrees rooted at "root"

    # 3. Make the inverted right subtree the left subtree of "root" and vice versa

```

Scaffolding Example

Given the root of a binary tree, invert the tree, and return its root.

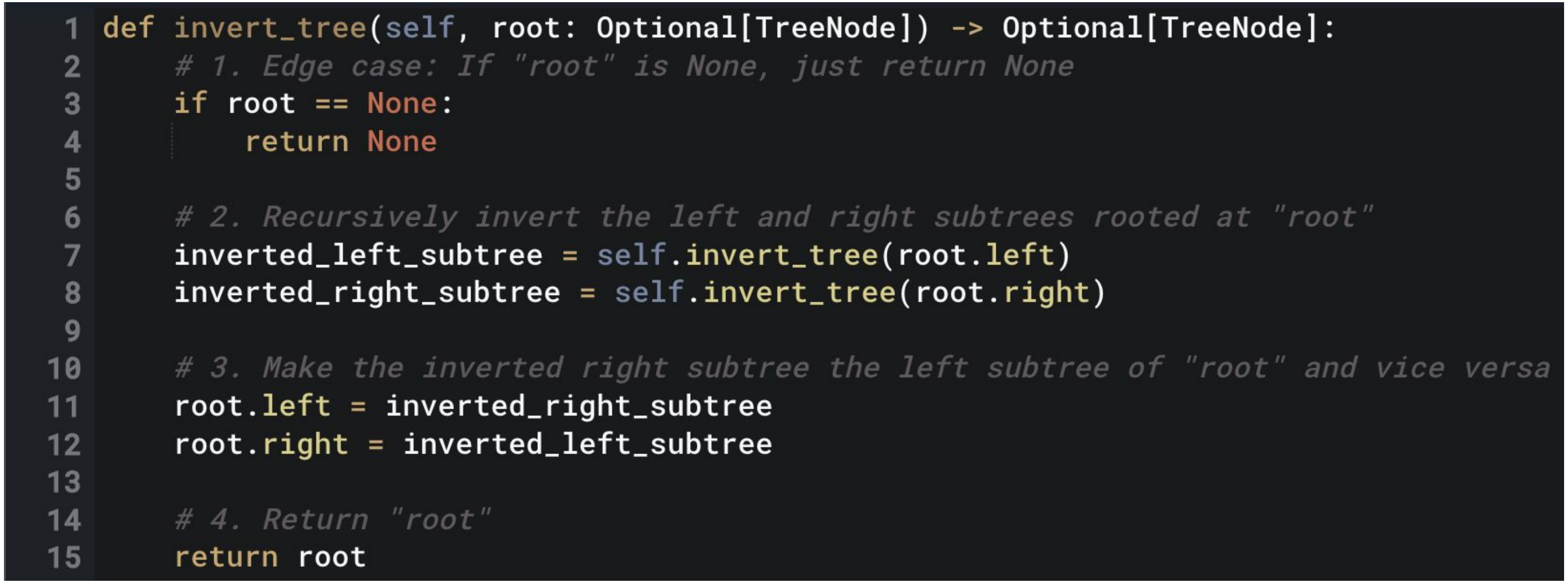


```python
def invert_tree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
    # 1. Edge case: If "root" is None, just return None

    # 2. Recursively invert the left and right subtrees rooted at "root"

    # 3. Make the inverted right subtree the left subtree of "root" and vice versa

    # 4. Return "root"

```

We can now fill in the blanks to get the desired code :)

Scaffolding Example

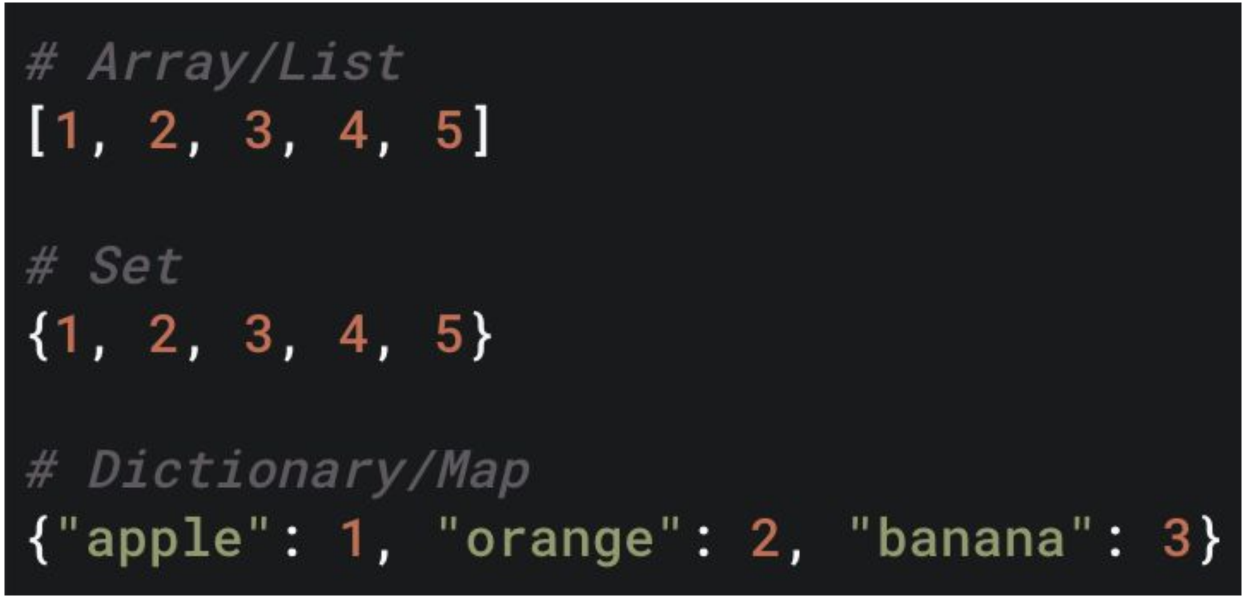
Given the root of a binary tree, invert the tree, and return its root.



```python
def invert_tree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
    # 1. Edge case: If "root" is None, just return None
    if root == None:
        return None

    # 2. Recursively invert the left and right subtrees rooted at "root"
    inverted_left_subtree = self.invert_tree(root.left)
    inverted_right_subtree = self.invert_tree(root.right)

    # 3. Make the inverted right subtree the left subtree of "root" and vice versa
    root.left = inverted_right_subtree
    root.right = inverted_left_subtree

    # 4. Return "root"
    return root
```

Drawing Diagrams

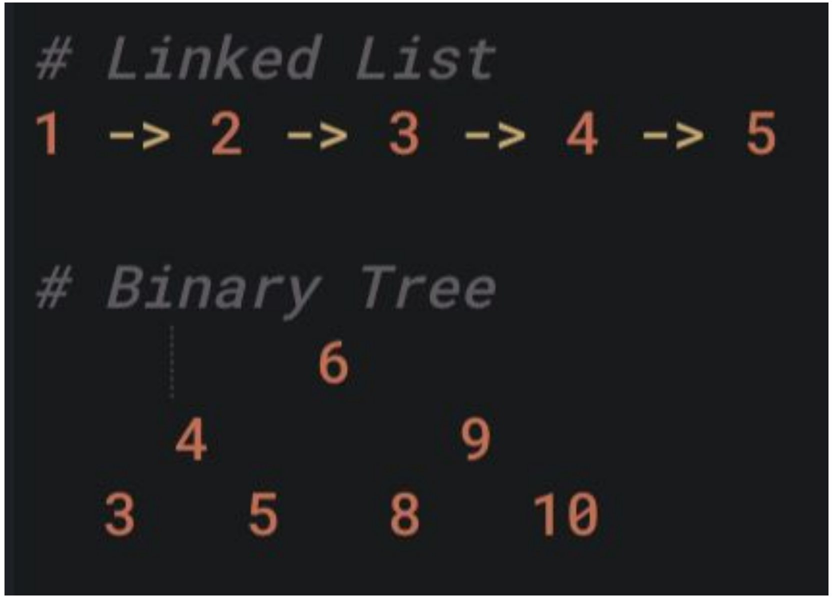
Note that the following are just **examples**; feel free to tailor/draw them in whatever way(s) you're comfortable with.



```
# Array/List
[1, 2, 3, 4, 5]

# Set
{1, 2, 3, 4, 5}

# Dictionary/Map
{"apple": 1, "orange": 2, "banana": 3}
```

Drawing Diagrams

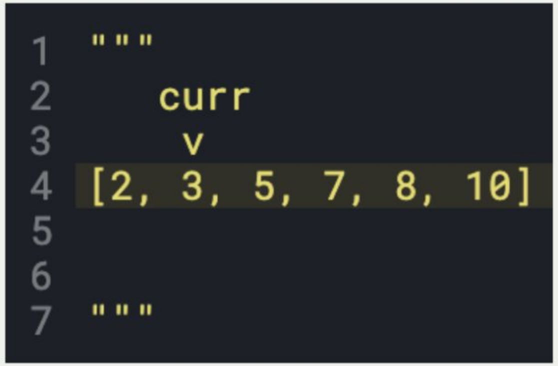
Note that the following are just **examples**; feel free to tailor/draw them in whatever way(s) you’re comfortable with.



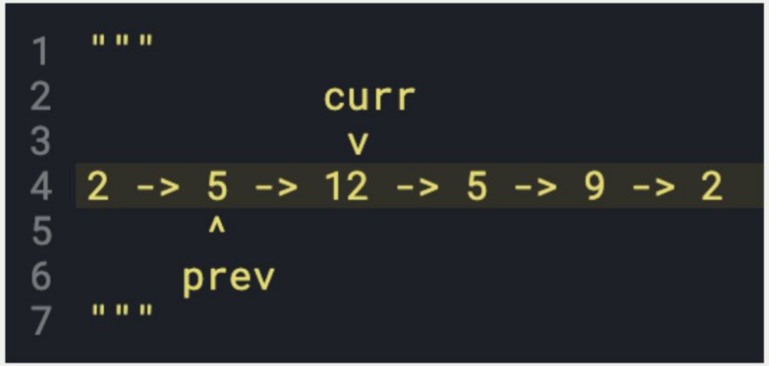
```
# Linked List
1 -> 2 -> 3 -> 4 -> 5

# Binary Tree
          6
    4           9
  3   5     8     10
```

Drawing Diagrams

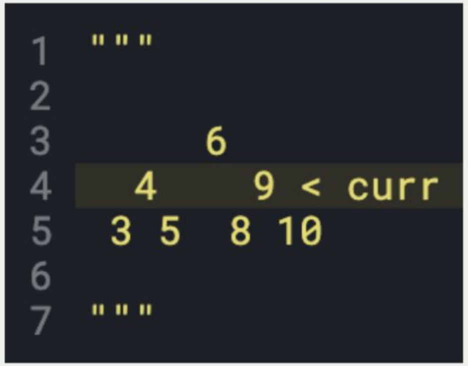
You can even add “pointers” for additional clarity.



```
"""
   curr
    v
[2, 3, 5, 7, 8, 10]


"""
```



```
"""
          curr
           v
2 -> 5 -> 12 -> 5 -> 9 -> 2
     ^
    prev
"""
```



```
"""

     6
  4    9 < curr
 3 5  8 10

"""
```

Outline for Tonight

- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

Goal

Coding is only part of the battle.

You should try to engage your interviewer as you code.

At the same time, you want to ensure that what you're coding is correct.

Overview

There are five key techniques to help you verbalise your thoughts appropriately:

1. See Interviewers as Collaborators
2. Keep Calm and Think Out Loud
3. Ask Questions, Not For Hints
4. Don't Rush!
5. Reiterate the Bounds!

1. See Interviewers as Collaborators

You should be solving the question **together**.

The interviewer will be assessing how well you work in a team (i.e. “would I want to be colleagues with XXX in future”?)

How you communicate during the interview will give off a lot of positive/negative signals

1. See Interviewers as Collaborators

Use "we" instead of "I".

E.g. "If we did a breadth first search, we'll get an answer in O(V + E) time”.

1. See Interviewers as Collaborators

Look up from the code editor and talk to the interviewer.

See how the interviewers are responding to your ideas and code.

You might even be able to catch some non-verbal cues on whether you are on the right track!

2. Keep Calm and Think Out Loud

If you're not talking, the interviewer will not know what you're thinking.

Tell the interviewer when you're not sure.

Let them know what you thought might have worked, and why you realised it doesn't.

2. Keep Calm and Think Out Loud

- Show the interviewer how you're tackling the problem and that you're not stuck
- Verbalise each step of your solution
- As you step through your code, you're more likely to catch any mistakes that you might have made

3. Ask Questions, Not For Hints

Don't pretend to know something when you don't.

- Instead say, "I'm not sure, but I would guess ____ here because ____."
- The "because" is important - it’s an opportunity to rule out other options with poor tradeoffs, or by pulling from other languages or problems.

3. Ask Questions, Not For Hints

It's ok to ask questions.

- Questions aren't just for the first 5 minutes of the interview.
- Ask questions that help identify the path forward.
- It's NEVER a good idea to ask, "Can you give me a hint?"

4. Don't Rush!

It's natural to code a bit slower if you have to talk while coding.

The time suggestion to finish your code in 10min is a guideline, not a rule.

4. Don't Rush!

Remember, a good interviewer will stretch your mind to see the limits of what you know.

Some problems are designed to take more time, and have natural stretch opportunities baked in.

4. Don't Rush!

Go ahead at a methodical pace so that you don’t commit careless mistakes and so that you can thoroughly think through the problem out loud.

- Chances are, you’ll finish the problem in less time and with fewer mistakes.

5. Reiterate the Bounds!

Don’t forget to communicate the time- and space-complexity of your algorithm! With your code finished, summarize the Big-O of your solution and why it’s less than ideal (if necessary).

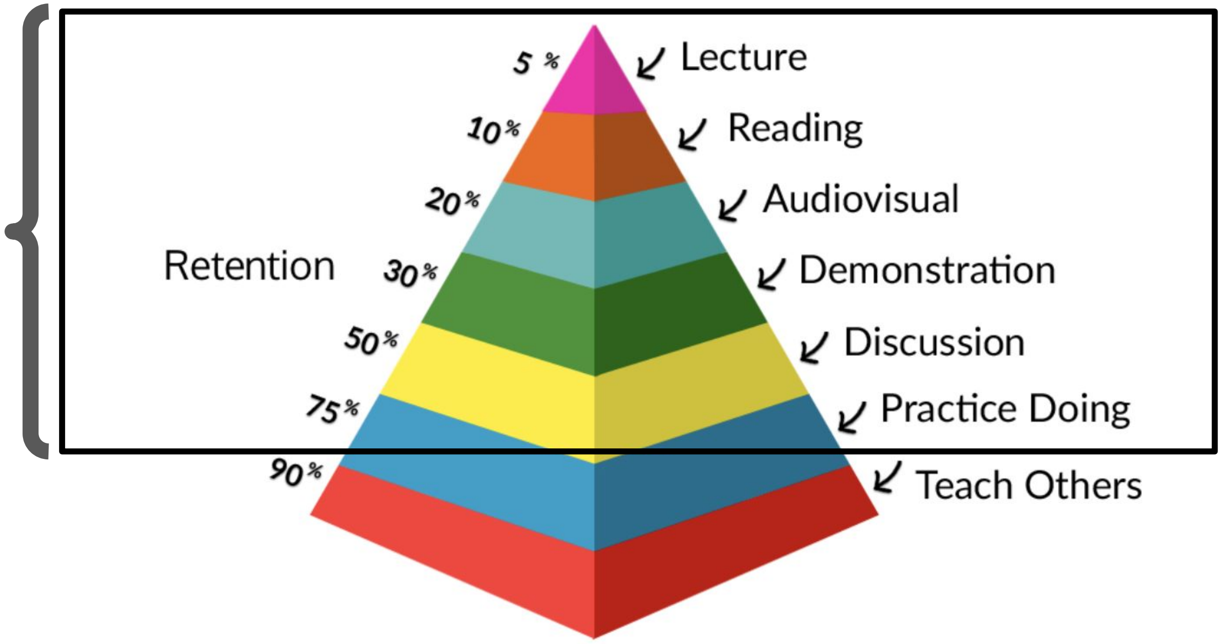
- Annotate chunks of your code with the various time and space complexities to demonstrate your understanding of the code.
- Explain trade offs with respect to time- and space-complexity in your current approach versus alternative approaches.

Talking As You're Coding (Summary)

1. See Interviewers as Collaborators
2. Keep Calm and Think Out Loud
3. Ask Questions, Not For Hints
4. Don't Rush!
5. Reiterate the Bounds!

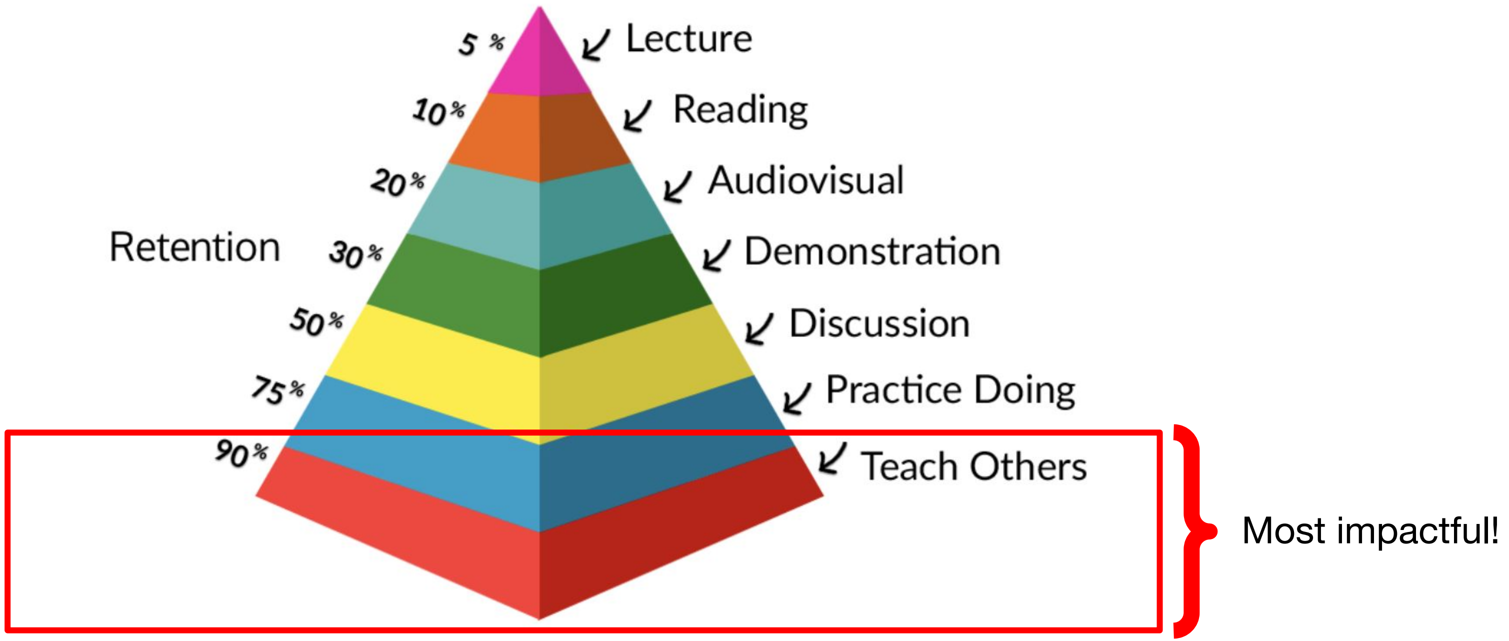
Building Proficiency (Secret Sauce)

Leetcode, mock interviews, etc.



Most people stop at the practice level.

Building Proficiency (Secret Sauce)



The key to hone your skills to the next level is to **teach others**!

Building Proficiency (Secret Sauce)

Examples of teaching others:

- Explain data structures and algorithms concepts to others, and make sure they **understand** what you’re trying to teach.
- Explain your Leetcode solutions to others, again with an emphasis on **understanding** from the other party.
- Become a teaching assistant, especially for CS2040/C/S (not applicable to everyone but really does wonders for technical interviews).

Building Proficiency (Secret Sauce)

Notice that the previous examples sound very similar to what you're doing during mock interviews?

True proficiency: You can explain complex topics **simply**.

By teaching TIPS, we (the teaching staff) are also building our proficiencies.

Outline for Tonight

- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

Outline for Tonight

- Admin Stuff
- Maximising Collaborative Editors
- Talking While Coding
- Demo With Hanming
- Language-Specific Workshops (CPP/Java/Python)

Language-Specific Workshops

```
switch (preferredLanguage) {
    case Python: Stay in main session
    case Java: https://bit.ly/tips-2022-java
    case C++: https://bit.ly/tips-2022-cpp
    default: Pick one! Choose your most comfortable language :)
}
```