CSC 148

Week 8

Sadia ‘Rain’ Sharmin

Classes begin 10 minutes after the hour

This Week's Cool Programmer(s) Feature: *Roy Clay*

Week 8

Trees

SUBTOPICS:

- Non-mutating Tree Methods
- Tree Traversals
- Tree Deletion Algorithms

Practice with Trees

From nature

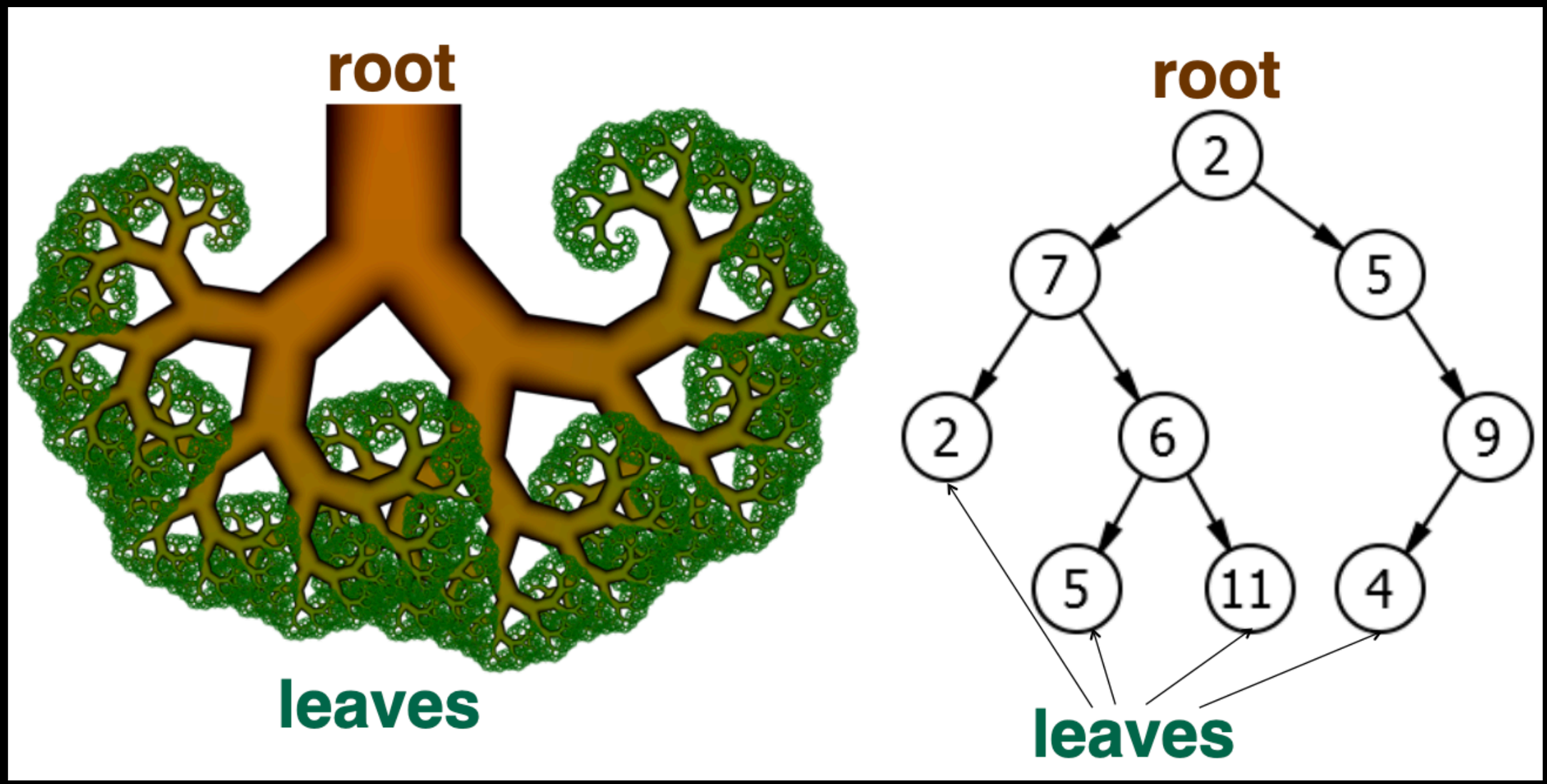
Recursion

From nature

Recursion

From nature to code

Recursion



Trees

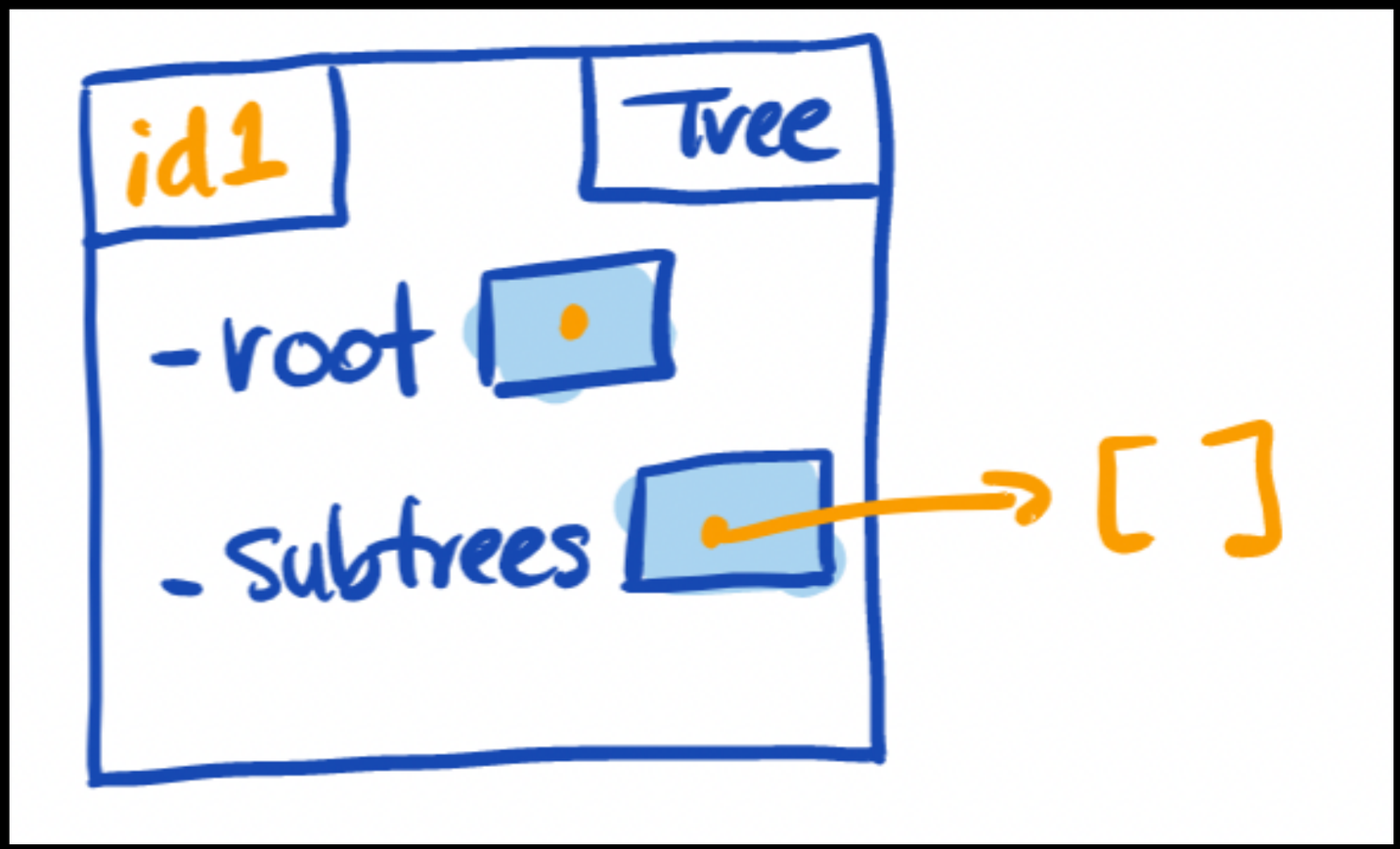
Definition

A tree is either empty, or

a root value connected to zero or more other trees, called the tree's subtrees

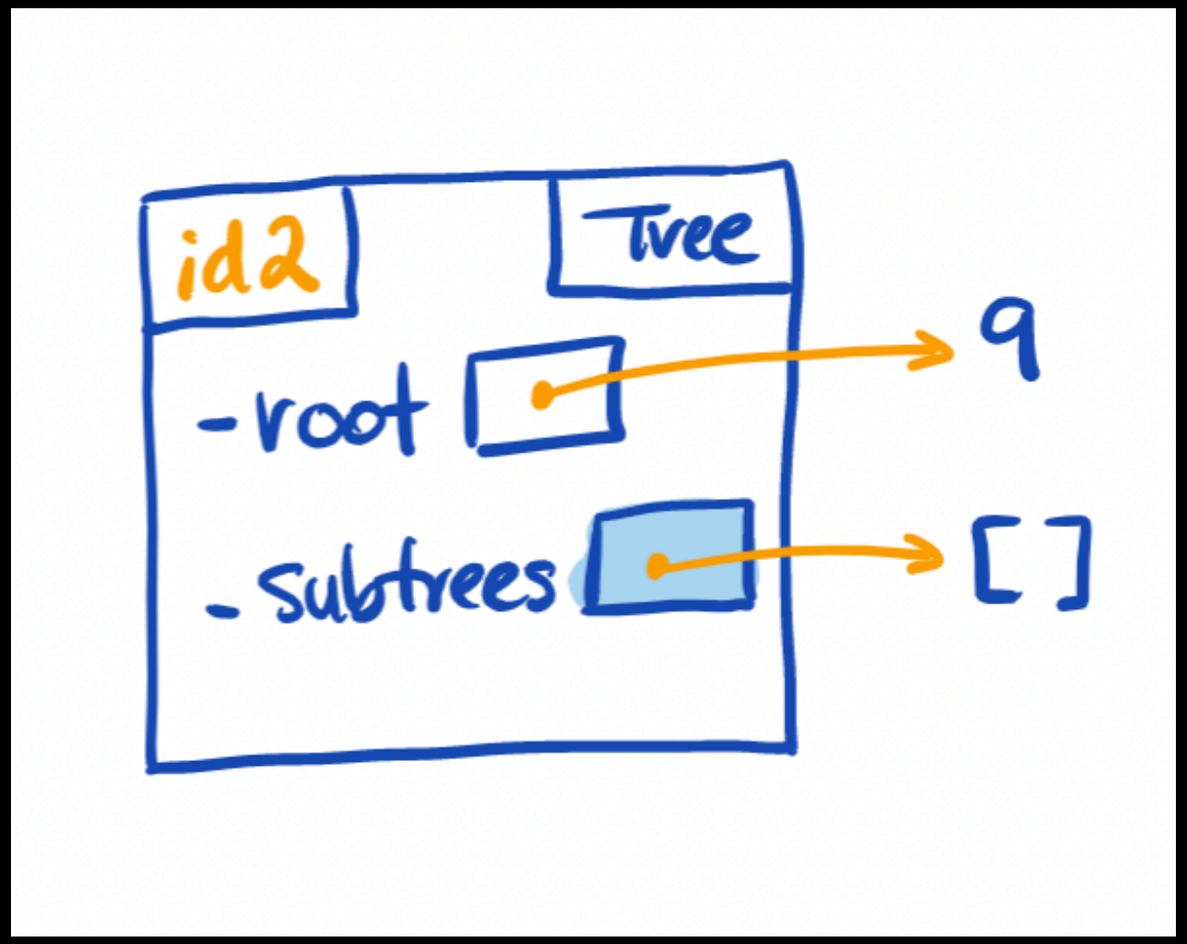
Our implementation

Empty tree:



Our implementation

Single value:

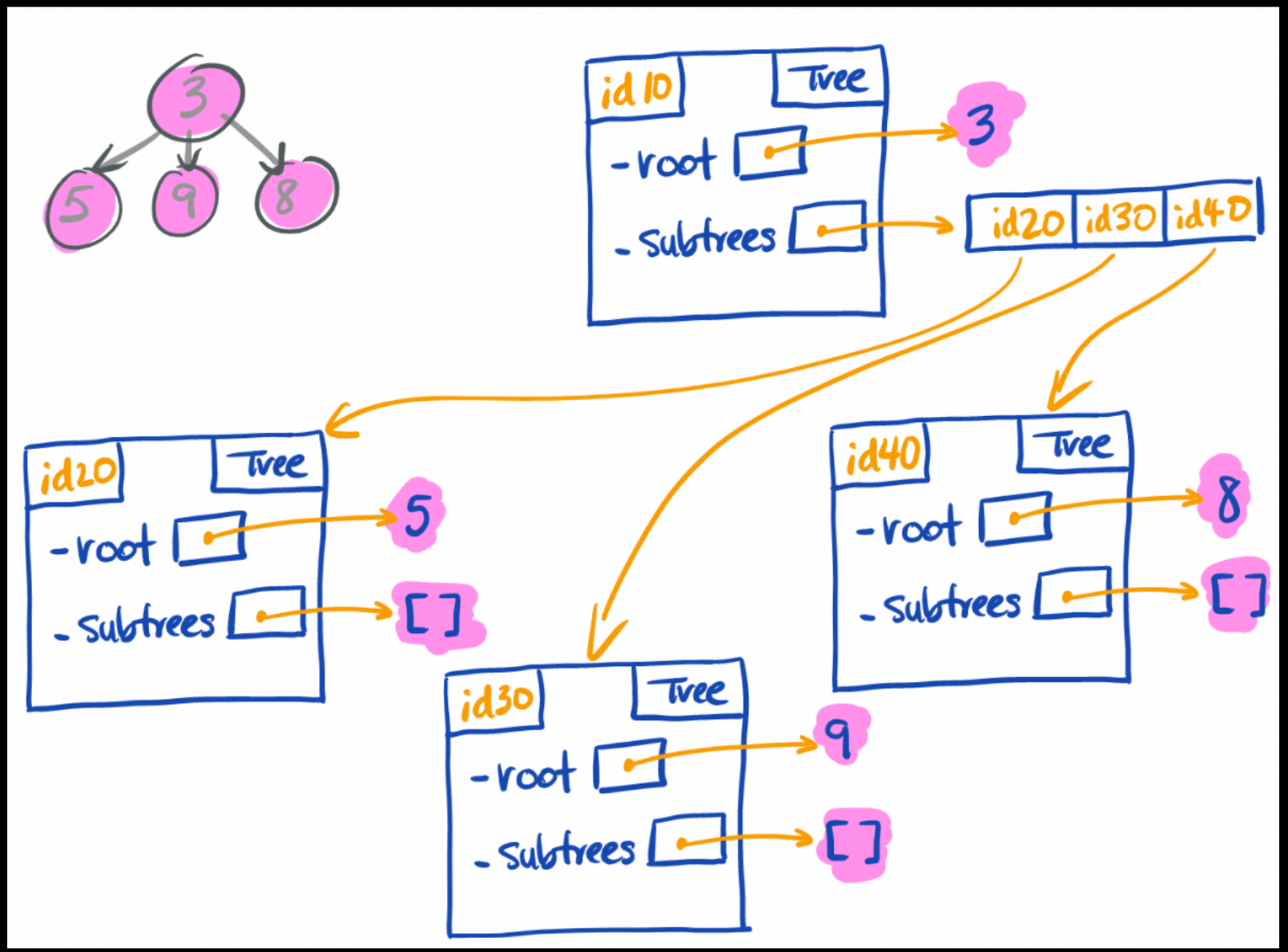


Trees

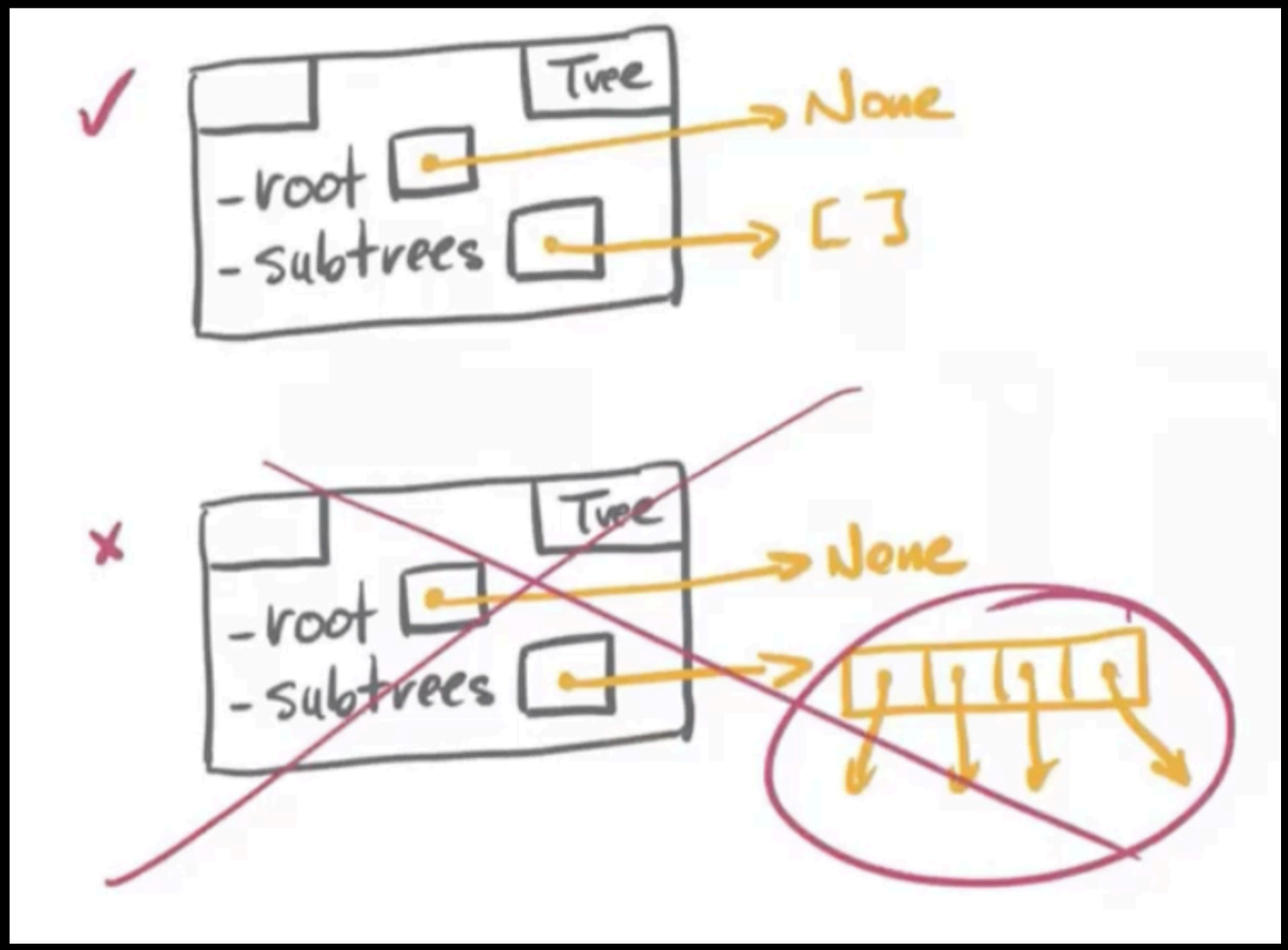
Trees

Our implementation

Multiple values:



When can _root be None?



If _root is None, our _subtrees attribute MUST be [].

Empty tree representation

Why not just say that t = None is an empty tree, instead of the weird t._root = None and t._subtrees = [] representation?

Because it cleans up our code for many tree methods/functions later on – buys us convenience.

Trees

```
def example_1(t: Tree)
    t.tree_method()
    # we can safely do this without checking if t is None

def example_2(forest: List[Tree]):
    for t in forest:
        t.treeMethod()  # same as above, this will never crash due to t being None
```

Trees

Activity

When asked to join the tree:

Stand up. You are the root of your own tree.

Ask 0, 1, or 2 other people who are still **seated** to be a root of their own tree.

These people are your subtrees, keep track of them!

Trees

Attributes and recursive template

```python
class Tree:
    """A recursive tree data structure.
    """
    # === Private Attributes ===
    # The item stored at this tree's root, or None if the tree is empty.
    _root: Optional[Any]
    # The list of all subtrees of this tree.
    _subtrees: List[Tree]
```

```python
def f(self) -> ...:
    if self.is_empty():          # tree is empty
        ...
    elif self._subtrees == []:   # tree is a single value (may be a redundant base case already covered)
        ...
    else:                        # tree has at least one subtree
        ...
        for subtree in self._subtrees:
            ... subtree.f() ...
        ...
```

Recursion

General strategy

1. Make a concrete example of each case that may be relevant.
2. For each of the recursive case(s):

   Write down the recursive calls and what they should return and/or mutate.

   Assume each will do what it should.

   Determine how to combine their results for the overall result.
3. See if any cases can collapse down.
4. Write the code.

WORKSHEET

Practice writing non-mutating Tree methods

Tree Traversals

Traversals

Different strategies

There are different strategies for tree traversal which involve visiting the tree's nodes in different orders.

We will discuss two today:

- Preorder
- Postorder

Traversals

Preorder

In preorder traversal, we deal with the root before the subtrees ("pre" because the root goes before)

Recursive Idea – Preorder traversal works as follows:

- do something with the node's value, e.g., print it
- visit its first subtree in preorder
- visit its second subtree in preorder

...

- visit its last subtree in preorder

See preorder function in tree_starter_code.py

Postorder

In postorder traversal, we deal with the root after the subtrees ("post" because the root goes after)

Recursive Idea – Postorder traversal works as follows:

- visit its first subtree in postorder
- visit its second subtree in postorder

...

- visit its last subtree in postorder
- do something with the node's value, e.g., print it

At-home exercise: Write a postorder tree traversal function

Traversals

Example

Let's take a look at our _str_indented method from the readings.

Which type of traversal do you think this uses?

Tree Deletion

Delete an item

```
def delete_item(self, item) -> bool:
    """Delete *one* occurrence of <item> from this tree.

    Return True if <item> was deleted, and False
    otherwise.

    """
```

Tree

Let's think about this

What to do if tree is empty?

If it has one node?

Multiple nodes? What if we're deleting the root? What if we're deleting something in one of the subtrees?

See delete_item_thoughts.pdf

Recursion

A start of an implementation

```
def delete_item(self, item) -> bool:
    if self.is_empty():
        return False  # item isn't in the tree
    elif self._root == item:
        self._delete_root()
        return True   # item was deleted
    else:
        ... # recurse somehow
```

Recursion

The problem of empty trees

```
>>> t = Tree(10, [Tree(1, []), Tree(2, []),
Tree(3, [])])
>>> t.delete_item(1)
True
>>> t.delete_item(2)
True
>>> t.delete_item(3)
True
```

A hidden assumption

“self._subtrees doesn’t contain any empty trees.”

Recursion

A representation invariant!

State assumptions using a RI:

"self._subtrees doesn't contain any empty trees."

WORKSHEET

Tree Deletion

Quizizz time!

Recursion and Trees

https://quizizz.com/admin/quiz/62263128bf562d001d15def6/recursion-and-trees

Practice

Nested list representation of trees

At-home

Draw diagrams of trees and lists to visualize the relationship

WORKSHEET

At-home Practice:

Trees and Nested Lists