# Mutable Trees

# Announcements

# String Representations

# String Representations

In Python, all objects produce two string representations:

- The **str** is (often) legible to **humans** & shows up when you **print**
- The **repr** is (often) legible to **Python** & shows up when you **evaluate** interactively

The **str** and **repr** strings are often the same, but not always

```
>>> from fractions import Fraction
>>> half = Fraction(1, 2)
>>> str(half)
'1/2'
>>> repr(half)
'Fraction(1, 2)'
>>> print(half)
1/2
>>> half
Fraction(1, 2)
```

If a type only defines a repr string, then the repr string is also the str string.

(Demo)

# Tree Class

# Tree Class

A Tree has a label and a list of branches; each branch is a Tree

```
class Tree:
    def __init__(self, label, branches=[]):
        self.label = label
        for branch in branches:
            assert isinstance(branch, Tree)
        self.branches = list(branches)
```

```
def tree(label, branches=[]):
    for branch in branches:
        assert is_tree(branch)
    return [label] + list(branches)
def label(tree):
    return tree[0]
def branches(tree):
    return tree[1:]
```

```
def fib_tree(n):
    if n == 0 or n == 1:
        return Tree(n)
    else:
        left = fib_tree(n-2)
        right = fib_tree(n-1)
        fib_n = left.label + right.label
        return Tree(fib_n, [left, right])
```

```
def fib_tree(n):
    if n == 0 or n == 1:
        return tree(n)
    else:
        left = fib_tree(n-2)
        right = fib_tree(n-1)
        fib_n = label(left) + label(right)
        return tree(fib_n, [left, right])
```

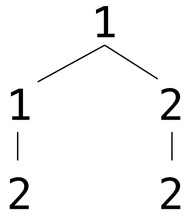
# Tree Practice

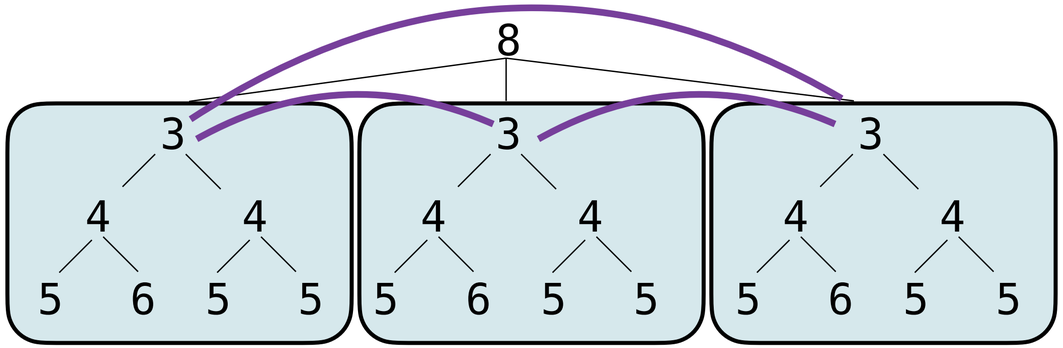
# Example: Count Twins

Implement twins, which takes a Tree t. It return the number of pairs of sibling nodes whose labels are equal.

```
def twins(t):
    """Count the pairs of sibling nodes with equal labels.

    >>> t1 = Tree(3, [Tree(4, [Tree(5), Tree(6)]), Tree(4, [Tree(5), Tree(5)])])
    >>> twins(t1)  # 4 and 5
    2
    >>> twins(Tree(1, [Tree(1, [Tree(2)]), Tree(2, [Tree(2)])]))
    0
    >>> twins(Tree(8, [t1, t1, t1]))  # 3 pairs of twins at the top, plus 2 in each branch
    9
    """
    count = 0
    n = len(t.branches)
    for i in range(n-1):
        for j in range(i+1, n):
            if t.branches[i].label == t.branches[j].label:
                count += 1
    return count + sum([twins(b) for b in t.branches])
```



# String Representation of Tree Class

(Demo)

https://code.cs61a.org/

# Example: make_even

# Example: largest_of_subtree

# Example: keep_k_largest