A Tree has a Label value 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 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)

Example: Pruning Trees
Removing subtrees from a tree is called pruning
Prune branches before recursive processing

def prune(t, n):
    # prune sub-trees whose label value is n.
    t.branches = [b for b in t.branches if b.label != n]
    for b in t.branches:
        prune(b, n)

(def)

Side Excursion: Equality
If x and y are two objects, the equality test, x == y, does not automatically mean what you want it to mean.
For example, Tree(4) != Tree(4) but after performing x = Tree(4), we do have x == x
The reason for this is that in Python,
• All values (conceptually, at least) are in fact pointers to objects, and
• By default, == on pointers compares the pointers themselves ("are these pointing at exactly the same object?").
• That is, by default == and != are the same as the is and is not operators.
• That can be changed on a class-by-class basis. For example, == on numbers, lists, tuples, strings, sets, and dictionaries means what we expect: the contents are the same.

Tree Mutation

Tree Class

A Tree has a Label value 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 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)

Tree Review
Recursive description (wooden trees):
A tree has a Label value and a list of branches
Each branch is a tree
A tree with zero branches is called a leaf

Relative description (family trees):
Each location in a tree is called a node
Each node has a value
One node can be the parent/child of another
Top node of tree is its root
Example: Pruning Trees

Removing subtrees from a tree is called pruning.

Prune branches before recursive processing.

E.g., want to prune (previously memorized) values.

Memoization:
- Returned by fib
- Found in cache
- Skipped

Memoization:

Example:

```
1 2 4 8 16 32 64 128 10 20 5
```

Hailstone Trees

Pick a positive integer n as the start.
If n is even, divide it by 2.
If n is odd, multiply it by 3 and add 1.
Continue this process until n is 1.

```python
def hailstone_tree(k, n=1):
    """Return a Tree in which the paths from the leaves to the root are all possible hailstone
    sequences of length k ending in n."""
```

```
1 2 4 8 16 32 64 10 20 3
```

(Hailstone Trees)