Composition

Announcements

Linked Lists

Linked List Structure

A linked list is either empty or a first value and the rest of the linked list

```python
Link(3, Link(4, Link(5, Link.empty)))
```

The first (zeroth) element is an attribute value

The rest of the elements are stored in a linked list

Linked List Class

```python
class Link:
    empty = ()

def __init__(self, first, rest=empty):
    assert rest is Link.empty or isinstance(rest, Link)
    self.first = first
    self.rest = rest
```

No method calls!

Property Methods

In some cases, we want the value of instance attributes to be computed on demand

For example, if we want to access the second element of a linked list

```python
>>> x = Link(3, Link(4, Link(5)))
>>> x.second
4
>>> x.second = 6
>>> x.second
6
```

A @property decorator on a method designates that it will be called whenever it is looked up on an instance

A @property.setter decorator on a method designates that it will be called whenever that attribute is assigned. <attribute> must be an existing property method.

Demo
TreeNode Class

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

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

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

(Demo)