Composition

Announcements

Linked Lists

Linked List Structure

Example: Range, Map, and Filter for Linked Lists

Linked List Class

Linked List Processing
Linked Lists Mutation

```python
s = Link(1, Link(2, Link(3)))
s.first = 5
s.rest = s.rest.rest.rest.rest.first
```

Linked Lists Can Change

Attribute assignment statements can change first and rest attributes of a Link

The rest of a linked list can contain the linked list as a sub-list

```
>>> s = Link(1, Link(3, Link(5)))
Notion: The actual environment diagram is much more complicated.
```

Adding to an Ordered List

```
def add(s, v):
    """Add v to an ordered list s with no repeats, returning modified s.""
    (Note: If v is already in s, then don't modify s, but still return it.)
    add(s, 0)
```

Adding to a Set Represented as an Ordered List

```
def add(s, v):
    """Add v to s, returning modified s.""
```

Adding to an Ordered List

```
def add(s, v):
    """Add v to an ordered list s with no repeats...""
    add(s, 0)
```

Adding to an Ordered List

```
def add(s, v):
    """Add v to an ordered list s with no repeats...""
    add(s, 0)
```

Adding to a Set Represented as an Ordered List

```
assert s is not List.empty
if s.first > v:
    s.first, s.rest = s.first, s.rest
elif s.first < v and empty(s.rest):
    s.rest = Link(v)
else:
    s.first = v:
    add(s.rest, v)
return s
```

```
def add(s, v):
    """Add v to s, returning modified s.""
>>> s = Link(1, Link(3, Link(5)))
>>> add(s, 4)
>>> s
```

Linked List Mutation Example

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to a Set Represented as an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```

Adding to an Ordered List

```
```
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 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])

Tree Abstraction (Review)

Recursive description (wooden trees):
A tree has a root label 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 label that can be any value.
The top node is the root node.

Example: Pruning Trees

Removing subtrees from a tree is called pruning.
Prune branches before recursive processing.

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

Tree Mutation

Example: Pruning Trees

Removing subtrees from a tree is called pruning.
Prune branches before recursive processing.

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