Composition
Announcements
Linked Lists
A linked list is either empty or a first value and the rest of the linked list.

A linked list is a pair. The first (zeroth) element is an attribute value. The rest of the elements are stored in a linked list.

A class attribute represents an empty linked list.

Link instance

<table>
<thead>
<tr>
<th>first:</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rest:</td>
<td></td>
</tr>
</tbody>
</table>

Link instance

<table>
<thead>
<tr>
<th>first:</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>rest:</td>
<td></td>
</tr>
</tbody>
</table>

Link instance

<table>
<thead>
<tr>
<th>first:</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>rest:</td>
<td></td>
</tr>
</tbody>
</table>

Link instance

| Link.empty |

Link(3, Link(4, Link(5, Link.empty)))
Linked List Structure

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

3, 4, 5

\[
\text{Link}(3, \text{Link}(4, \text{Link}(5, \text{Link}.\text{empty})))
\]
Linked List Class

Linked list class: attributes are passed to \texttt{\_\_init\_}

class \texttt{Link}:

\begin{itemize}
  \item \texttt{empty} = ()
  \item \texttt{def \_\_init\_(self, first, rest=empty):}
    \begin{itemize}
      \item \texttt{assert rest is Link.empty or isinstance(rest, Link)}
      \item \texttt{self.first = first}
      \item \texttt{self.rest = rest}
    \end{itemize}
\end{itemize}

\texttt{help(isinstance)}: Return whether an object is an instance of a class or of a subclass thereof.

\begin{verbatim}
Link(3, Link(4, Link(5)))
\end{verbatim}

(Demo)
Property Methods
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
>>> s = Link(3, Link(4, Link(5)))
>>> s.second
4
>>> s.second = 6
>>> s.second
6
>>> s
Link(3, Link(6, Link(5)))
```

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

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

(Demo)
Tree Class
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. A tree starts at the root.

Relative description (family trees):
Each location in a tree is called a node. Each node has a label that can be any value. One node can be the parent/child of another. The top node is the root node.

People often refer to labels by their locations: "each parent is the sum of its children"
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])

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:]