The Closure Property of Data Types

- A method for combining data values satisfies the closure property if:
- The result of combination can itself be combined using the same method
- Closure is powerful because it permits us to create hierarchical structures
- Hierarchical structures are made up of parts, which themselves are made up of parts, and so on

Lists can contain lists as elements (in addition to anything else)

Box-and-Pointer Notation in Environment Diagrams

Lists are represented as a row of index-labeled adjacent boxes, one per element
Each box either contains a primitive value or points to a compound value

Slicing

Slicing Creates New Values

Sequence Aggregation

Several built-in functions take iterable arguments and aggregate them into a value
- \( \text{sum} \) (iterable) -> value
  - Return the sum of an iterable of numbers (not strings) plus the value of parameter 'start' (which defaults to 0). When the iterable is empty, return start.
- \( \text{max} \) (iterable) -> value
  - Return the largest item in the iterable.
- \( \text{all} \) (iterable) -> bool
  - Return True if all elements of the iterable are truthy; return False if the iterable is empty.
Tree Abstraction

A function that creates a tree from another tree is typically also recursive:

```python
def tree(label, branches=[]):
    if is_leaf(tree):
        return label[
            tree[branches]]
    else:
        branches, [count_leaves(b) for][[branches]]
        return sum(count_leaves(branch) for branch in branches(tree))

Example: Printing Trees

>>> tree([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])
([1, 0, 1, 0, 1, 1, 0, 1])
```

Tree Processing Uses Recursion

Processing a leaf is often the base case of a tree processing function:

```python
def count_leaves(tree):
    """Count the leaves of a tree."""
    if is_leaf(tree):
        return 1
    else:
        return sum(count_leaves(b) for branch in branches(tree))
```

Discussion Question

Implement leave() which returns a list of the leaf labels of a tree:

```python
def leave(tree):
    """Return a list containing the leaves of tree."
    if is_leaf(tree):
        return [label(tree)]
    else:
        return [label(b) for branch in branches(tree) if is_leaf(b)]
```