Suppose we define a data structure called a binary tree. A tree is either empty (it’s None), or it is a Tree “node” that holds a value and contains references to two other (possibly empty) subtrees.

class Tree(object):
    __slots__ = [ "_value", "_left", "_right" ]
    def __init__(self, value, left=None, right=None):
        self._value = value # the data stored in this tree node
        self._left = left  # left is None, or a Tree
        self._right = right # right is None, or a Tree

@property
def value(self): # the t.value property
    return self._value

@property
def left(self): # the t.left property
    return self._left

@property
def right(self): # the t.right property
    return self._right

Trees are acyclic; the left and right subtrees do not overlap; the subtrees are disjoint.

1. Write a recursive method of Tree that computes the number of data values stored in a Tree.

    def __len__(self): # hint: it’s at least one!
        """Returns the number of values stored in the tree."""

2. Write a recursive method of Tree that computes the sum of the values in the tree:

    def sum(self):
        """Returns the sum of all the values found in this tree."""