Daily Coding Problem #3

Problem

This problem was asked by Google.

Given the root to a binary tree, implement `serialize(root)`, which serializes the tree into a string, and `deserialize(s)`, which deserializes the string back into the tree.

For example, given the following `Node` class

```
class Node:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

The following test should pass:

```
node = Node('root', Node('left', Node('left.left')), Node('right'))
assert deserialize(serialize(node)).left.left.val == 'left.left'
```

Solution

There are many ways to serialize and deserialize a binary tree, so don't worry if your solution differs from this one. We will be only going through one possible solution.

We can approach this problem by first figuring out what we would like the serialized tree to look like. Ideally, it would contain the minimum information required to encode all the necessary information about the binary tree. One possible encoding might be to borrow S-expressions from Lisp. The tree `Node(1, Node(2), Node(3))` would then look like '(1 (2 () ()) (3 () ()))', where the empty brackets denote nulls.

To minimize data over the hypothetical wire, we could go a step further and prune out some unnecessary brackets. We could also replace the 2-character '()' with '#'. We can then infer leaf nodes by their form 'val # #' and thus get the structure of the tree that way. Then our tree would look like `1 2 # # 3 # #`.

```
def serialize(root):
    if root is None:
        return '#'
    return '{} {} {}'.format(root.val, serialize(root.left), serialize(root.right))

def deserialize(data):
    def helper():
        val = next(vals)
        if val == '#':
            return None
        node = Node(int(val))
        node.left = helper()
        node.right = helper()
        return node
    vals = iter(data.split())
    return helper()
```

This runs in O(N) time and space, since we iterate over the whole tree when serializing and deserializing.