Daily Coding Problem #50

Problem

This problem was asked by Microsoft.

Suppose an arithmetic expression is given as a binary tree. Each leaf is an integer and each internal node is one of '+', '−', '∗', or '/'.

Given the root to such a tree, write a function to evaluate it.

For example, given the following tree:

```
    *
   / \
  +    +
 / \  / \
3  2  4  5
```

You should return 45, as it is (3 + 2) * (4 + 5).

Solution

This problem should be straightforward from the definition. It will be recursive. We check the value of the root node. If it's one of our arithmetic operators, then we take the evaluated value of our node's children and apply the operator on them.

If it's not an arithmetic operator, it has to be a raw number, so we can just return that.

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

PLUS = "+"
MINUS = "-"
```

```
TIMES = "*"
DIVIDE = "/"
def evaluate(root):
    if root.val == PLUS:
        return evaluate(root.left) + evaluate(root.right)
    elif root.val == MINUS:
        return evaluate(root.left) - evaluate(root.right)
    elif root.val == TIMES:
        return evaluate(root.left) * evaluate(root.right)
    elif root.val == DIVIDE:
        return evaluate(root.left) / evaluate(root.right)
    else:
        return root.val
```

This runs in O(N) time and space.
