Daily Coding Problem #27

Problem

This problem was asked by Facebook.

Given a string of round, curly, and square open and closing brackets, return whether the brackets are balanced (well-formed).

For example, given the string "([])[]({})", you should return true.

Given the string "([)]" or "(((", you should return false.

Solution

In this case, it's easy to start with a simplified case of the problem, which is dealing with only round brackets. Notice that in this case, we just need to keep track of the current number of open brackets -- each closing bracket should be matched

with the rightmost open bracket. So we can keep a counter and increment it for every open bracket we see and decrement it on every closing bracket. If we get to the end of the string and have a non-zero number, then it means it's unbalanced. A negative number would indicate more closing brackets than open ones, and a positive number would indicate the opposite.

In the case of round, curly, and square brackets, we need to also keep track of what *kind* of brackets they are as well, because we can't match a round open bracket with a curly square. In this case, we can use a stack to keep track of the actual bracket character and push onto it whenever we encounter an open bracket, and pop if we encounter a matching closing bracket to the top of the stack. If the stack is empty or it's not the correct matching bracket, then we'll return false. If, by the end of the iteration, we have something left over in the stack, then it means it's unbalanced -- so we'll return whether it's empty or not.

```
def balance(s):
    stack = []
    for char in s:
        if char in ["(", "[", "{"]:
            stack.append(char)
        else:
            # Check character is not unmatched
            if not stack:
                return False

            # Char is a closing bracket, check top of stack if it matches
            if (char == ")" and stack[-1] != "(") or \
               (char == "]" and stack[-1] != "[") or \
               (char == "}" and stack[-1] != "{"):
                return False
            stack.pop()

    return len(stack) == 0
```

Fun fact: "(())" is not a palindrome, nor is "()()". "())(" is a palindrome, though.
