Daily Coding Problem #43

Problem

This problem was asked by Amazon.

Implement a stack that has the following methods:

- push(val), which pushes an element onto the stack
- pop(), which pops off and returns the topmost element of the stack. If there are no elements in the stack, then it should throw an error or return null.
- max(), which returns the maximum value in the stack currently. If there are no elements in the stack, then it should throw an error or return null.

Each method should run in constant time.

Solution

Implementing the stack part (push and pop) of this problem is easy -- we can just use a typical list to implement the stack with `append` and `pop`. However, getting the max in constant time is a little trickier. We could obviously do it in linear time if we popped off everything on the stack while keeping track of the maximum value, and then put everything back on.

We can use a secondary stack that *only* keeps track of the max values at any time. It will be in sync with our primary stack, as in it will have the exact same number of elements as

our primary stack at any point in time, but the top of the stack will always contain the maximum value of the stack.

We can then, when pushing, check if the element we're pushing is greater than the max value of the secondary stack (by just looking at the top), and if it is, then push that instead. If not, then maintain the previous value.

```
class MaxStack:
    def __init__(self):
        self.stack = []
        self.maxes = []

    def push(self, val):
        self.stack.append(val)
        if self.maxes:
            self.maxes.append(max(val, self.maxes[-1]))
        else:
            self.maxes.append(val)

    def pop(self):
        if self.maxes:
            self.maxes.pop()
        return self.stack.pop()

    def max(self):
        return self.maxes[-1]
```

Everything should run in O(1) time.