Daily Coding Problem #139

Problem

This problem was asked by Google.

Given an `iterator` with methods `next()` and `hasNext()`, create a wrapper iterator, `PeekableInterface`, which also implements `peek()`. `peek` shows the next element that would be returned on `next()`.

Here is the interface:

```
class PeekableInterface(object):
    def __init__(self, iterator):
        pass

    def peek(self):
        pass

    def next(self):
        pass

    def hasNext(self):
        pass
```

Solution

This problem can be solved by storing an instance variable `_next` in our class that holds the following invariant:

> `_next` always holds the next item that would be returned on`next()`.

We can follow this by setting `_next` to `next(self.iterator)` in the constructor and then updating on each `next()` call.

Using this invariant, we can then implement `peek` by simply returning `_next` and we can implement `hasNext` by checking that it's not `None`.

```
class PeekableInterface(object):
    def __init__(self, iterator):
        self.iterator = iterator
        self._next = next(self.iterator)

    def peek(self):
        return self._next

    def next(self):
        result = self._next
        self._next = next(self.iterator)
        return result

    def hasNext(self):
        return self._next is not None
```