Computer Science CS134C (Spring 2018)
Duane A. Bailey
Laboratory 6
Building an Oracle (due 11pm, Wednesday)

Objective. To build a simple class that generates text in an intelligent (?) manner.

This week we'll write a simple little class, an `Oracle`, that can be trained to generate "readable" random text. The class makes use of a technique that *fingerprints* a corpus by keeping track of a distribution of combinations of n letters.

The Tasks. Here are the steps to completing this week's lab.

1. Download the starter kit for this package in the usual manner:

   ```
   git clone ssh://18xyz@davey.cs.williams.edu/~dab/18xyz/lab6.git lab6
   ```

   This kit is fairly minimal; you'll be writing most of the code from scratch.

2. In a file `oracle.py` write a class, `Oracle`, that inherits from the `object` class. It would typically be used in the following manner:

   ```
   o = Oracle(window=3)
   text = ' '.join([line.strip() for line in open('tomsawyer.txt')])
   o.intern(text)
   print(o.generate('Tom'))
   ```

   The initializer for the `Oracle` takes a window size, `w`. When the `Oracle` scans a corpus of text, it scans the text from beginning to end keeping track of the frequency of each combination of `w` characters. We'll think of this, essentially, as a *fingerprint* of the corpus.

   The `generate` method, given a seed of at least `w-1` characters, generates a string of text, randomly, given the fingerprint internalized in the `Oracle`.

3. Write the `__init__` method. This method should accept a window size that defaults to 3. The distribution for the `Oracle` is initially uniform; there's no real fingerprint.

4. Write a method, `o.intern(s)`, that takes a string, `s`, and records the occurrences of `w` character combinations, where `w` is the window size of the `Oracle`.

5. Write a method, `o.follows(s)`. The string should be at least `w-1` characters long, where `w` is the `Oracle`'s window size. It returns a random character that, given knowledge of the interned text, is likely to follow `s`. For example, if the window size is 3, then

   ```
   >>> o = Oracle()
   >>> o.intern('this, that, and the other thing')
   >>> o.follows('th')
   'e'
   ```

another possibilities would be `'i'` or `'a'`. You should think about what happens if the seed string does not appear in the interned distribution.

6. Write a method, `o.generate(seed)`, that, given a string of at least length `w-1` generates a string that follows the distribution of the original corpus.

   The length of the string should be bounded by a default `length`. For example, length might be 80 if you're generating pages of lines of text. The length might be 140 if you wanted to use this to generate tweets.

7. To demonstrate the functionality of your `Oracle`, write code (guarded by an `if` __name__ statement), that prints a page of text motivated by the fingerprint of Twain's *Tom Sawyer*. Or you may use another text (I've included Jane Austen's *Pride and Prejudice*, if you're so inclined).

8. Feel free to add other features to the `Oracle`. Are there properties that might be interesting? How might you simplify the interning of text from a file like `tomsawyer.txt`?

9. Make sure that you provide appropriate documentation strings for the class and its methods.

Remember: use git to `add` and `commit` changes to files you add to this project.

⋆