Homework 4

1. Write a function called `mobysorted` that sorts a list of words according to their count in *Moby Dick,* from most frequent to least frequent. For example:

```
>>> mobysorted(['whale', 'anvil', 'the'])
['the', 'whale', 'anvil']
```

2. Write a function called `nvowels` that computes the number of vowels in a word. Define vowels to be *a, e, i, o,* and *u.* (Ignore any cases in which *y* is a vowel.) You should implement it using the `sum` function. Write a list comprehension in which you collect a 1 for each vowel, and call `sum` on the resulting list. An example:

```
>>> nvowels('tuesday')
3
```

3. Write a function called `vsort` that takes a list of words and sorts them by "voweliness," defining voweliness to be the PROPORTION of letters in the word that are vowels. Words should be sorted from most vowely to least vowely. For example:

```
>>> vsort(['test', 'a', 'iamb', 'aeolian'])
['a', 'aeolian', 'iamb', 'test']
```

Note that `'a'` is 100% vowely, whereas `'aeolian'` is 5/7 vowely.

4. Do item #15 of Handout 4. Specifically:

a. Define a function `diversity` that takes a text and returns the number of word types among the first 1000 tokens of the text. Before counting types, map the words to lowercase, but do not discard punctuation. An example:

```
>>> diversity(text1)
433
```

b. Construct a list called `texts` containing `text1`, `text2`, ..., `text9`. Then construct a list containing pairs (d, n), where n is the name of a text and d is its diversity. Sort the list from highest diversity to lowest diversity, and store the result in the variable `A4`.

Tip: by default, a list of pairs will be sorted by the first elements of the pairs, with ties broken by sorting according to the second element. For example:

```
>>> sorted([(3, 'dog'), (2, 'hi'), (3, 'cat')])
[(2, 'hi'), (3, 'cat'), (3, 'dog')]
```

It sorts by the numbers first, then alphabetically where there are ties.

5. Write a function called `zipfslope` that takes a text and returns the ratio of the count of the most-frequent word to the second-most-frequent word. If Zipf's law is exactly correct, the value will be 2. Example:

```
>>> zipfslope(['a', 'a', 'a', 'a', 'a', 'b', 'b', 'c'])
2.5
```

6. A consequence of Zipf's law is that most word types are rare.

a. A word that occurs only once in a text is called a *hapax legomenon,* the Greek expression for "read once." The `FreqDist` method `hapaxes` returns the list of items that have count one. Write a function called `hapax_prop` that takes a text as input, and returns the *proportion* of words that have count one. For example:

```
>>> hapax_prop(['a', 'a', 'b', 'b', 'c', 'c', 'd'])
.25
```

b. Create a list of pairs (p, n) where n is a text name and p is the hapax proportion. Sort the list from highest to lowest hapax proportion, and set `A6` to the result. You should also print it out readably for your own perusal.

7. Another consequence of Zipf's law is that most word *tokens* belong to high-frequency types.

a. Write a function called `cumulative` that takes two arguments: a frequency distribution and a number n. It should sum the frequencies of the n most-frequent items in the distribution. For example:

```
>>> fd = FreqDist(['a', 'a', 'a', 'b', 'b', 'c', 'd', 'e'])
>>> cumulative(fd, 2)
.625
```

The 2 most-frequent types have frequencies 3/8 and 2/8, for a total of $5/8 = .625$.

b. Write a function called `coverage20` that takes a text as input and returns the proportion of the text that is covered by the 20 most-frequent types.

c. Create a list of pairs (c, n) where n is a text name and c is the value of `coverage20` for the text. Sort from highest-coverage to lowest-coverage and store in `A7`. Print the table for your own perusal.

Discussion Questions

8. Are there surprises in the table of diversities?

9. Suppose a politician bemoans the fact that just 20 words account for more than a third of the text produced by kids these days. Is this an indication of our cultural demise?

10. Is `coverage20` a good way to discriminate between “low-brow” texts and “literary” texts?