nltk book: Language Processing and Python

• 2 A Closer Look at Python: Texts as Lists of Words: http://www.nltk.org/book/ch01.html

• Assuming sent1,..,sent9

• from nltk.book import *

```python
>>> sent1
['Call', 'me', 'Ishmael', '.']
>>> len(sent1)
4
>>> lexical_diversity(sent1)
1.0
```

```python
>>> sent2
['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sussex', '.']
>>> sent3
['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven', 'and', 'the', 'earth', '.']
```
nltk book: Language Processing and Python

• sent2, sent3

```python
>>> sent2
['The', 'family', 'of', 'Dashwood', 'had', 'long', 'been', 'settled', 'in', 'Sussex', '.']
>>> sent3
['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven', 'and', 'the', 'earth', '.']
``` 

• + for concatenation

```python
>>> ['Monty', 'Python'] + ['and', 'the', 'Holy', 'Grail']
['Monty', 'Python', 'and', 'the', 'Holy', 'Grail']
```
nltk book: Language Processing and Python

- `append()` to the end of the list (mutates the list)

```
>>> sent1.append("Some")
>>> sent1
['Call', 'me', 'Ishmael', '.', 'Some']
```

- `append()` vs. `extend()` to the end of the list:

```
x = [1, 2, 3]
x.append([4, 5])
print(x)
gives you: [1, 2, 3, [4, 5]]
```

```
x = [1, 2, 3]
x.extend([4, 5])
print(x)
gives you: [1, 2, 3, 4, 5]
```

from stackoverflow.com
nltk book: Language Processing and Python

• Indexing [<index>]:

```
>>> text4[173]
'awaken'
```

```
>>> text4.index('awaken')
173
```

• Slices [<index>:<index>]: (can omit either <index>, default value)

```
>>> text5[16715:16735]
['U86', 'thats', 'why', 'something', 'like', 'gamefly', 'is', 'so', 'good', 'because', 'you', 'can', 'actually', 'play', 'a', 'full', 'game', 'without', 'buying', 'it']
```

```
>>> text6[1600:1625]
['We', '..', 're', 'an', 'anarcho', '–', 'syndicalist', 'commune', '..', 'We', 'take', 'it', 'in', 'turns', 'to', 'act', 'as', 'a', 'sort', 'of', 'executive', 'officer', 'for', 'the', 'week']
```
We know indexing works on strings (as well as lists):

- Repetition (*), Concatenation (+):

```python
>>> name = 'Monty'
>>> name[0]
'M'
>>> name[:4]
'Mont'
```

- `.join() .split()`

```python
>>> name * 2
'MontyMonty'
>>> name + '!
'Monty!'
```

```python
>>> ' '.join(['Monty', 'Python'])
'Monty Python'
>>> 'Monty Python'.split()
['Monty', 'Python']
```
nltk book: Language Processing and Python

• Understanding check:

```python
>>> saying = ['After', 'all', 'is', 'said', 'and', 'done',
...            'more', 'is', 'said', 'than', 'done']
>>> tokens = set(saying)
>>> tokens = sorted(tokens)
>>> tokens[-2:]
what output do you expect here?
>>> 
```

**Answer:** Last two words by alphabetic sorting...
3.1 Frequency Distributions

```python
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G. K. Chesterton 1908

>>> fdist1 = FreqDist(text1)
>>> type(fdist1)
<class 'nltk.probability.FreqDist'>
>>> print(fdist1)
<FreqDist with 19317 samples and 260819 outcomes>
```
Table 3.1:

Functions Defined for NLTK's Frequency Distributions

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdist = FreqDist(samples)</td>
<td>create a frequency distribution containing the given samples</td>
</tr>
<tr>
<td>fdist[sample] += 1</td>
<td>increment the count for this sample</td>
</tr>
<tr>
<td>fdist['monstrous']</td>
<td>count of the number of times a given sample occurred</td>
</tr>
<tr>
<td>fdist.freq('monstrous')</td>
<td>frequency of a given sample</td>
</tr>
<tr>
<td>fdist.N()</td>
<td>total number of samples</td>
</tr>
<tr>
<td>fdist.most_common(n)</td>
<td>the n most common samples and their frequencies</td>
</tr>
<tr>
<td>for sample in fdist:</td>
<td>iterate over the samples</td>
</tr>
<tr>
<td>fdist.max()</td>
<td>sample with the greatest count</td>
</tr>
<tr>
<td>fdist.tabulate()</td>
<td>tabulate the frequency distribution</td>
</tr>
<tr>
<td>fdist.plot()</td>
<td>graphical plot of the frequency distribution</td>
</tr>
<tr>
<td>fdist.plot(cumulative=True)</td>
<td>cumulative plot of the frequency distribution</td>
</tr>
<tr>
<td>fdist1</td>
<td>= fdist2</td>
</tr>
<tr>
<td>fdist1 &lt; fdist2</td>
<td>test if samples in fdist1 occur less frequently than in fdist2</td>
</tr>
</tbody>
</table>
specifically relevant to Moby Dick; other reported words are generic

"English plumbing"
 nltk book: Language Processing and Python
• Extract long words (using list comprehension):

```python
>>> V = set(text1)
>>> long_words = [w for w in V if len(w) > 15]
>>> sorted(long_words)
['CIRCUMNAVIGATION', 'Physiognomically', 'apprehensiveness', 'cannibalistically',
'characteristically', 'circumnavigating', 'circumnavigation', 'circumnavigations',
'comprehensiveness', 'hermaphroditical', 'indiscriminately', 'indispensableness',
'irresistibleness', 'physiognomically', 'preternaturalness', 'responsibilities',
'simultaneity', 'subterraneousness', 'supernaturalness', 'superstitiousness',
'uncomfortableness', 'uncompromisedness', 'undiscriminating', 'uninterpenetratingly']
```
nltk book: Language Processing and Python

- text5: chat corpus
- Pick out all the words longer than 7 characters that occur more than 7 times (using list comprehension) and sort them:

```python
g>>> fdist5 = FreqDist(text5)
g>>> sorted([w for w in set(text5) if len(w) > 7 and fdist5[w] > 7])
['#14-19teens', '#talkcity_adults', '(((((((((', '........', 'Question', 'actually', 'anything', 'computer', 'cute.-ass', 'everyone', 'football', 'innocent', 'listening', 'remember', 'seriously', 'something', 'together', 'tomorrow', 'watching']
```
nltk book: Language Processing and Python

• Classes: FreqDist vs. Text

```python
>>> fdist1.freq('whale')
0.003473673313677301
>>> fdist1['whale']
906
>>> text1.count('whale')
906
>>> len(text1)
260819
>>> text1.count('whale') / len(text1)
0.003473673313677301
>>> ```
nltk book: Language Processing and Python

• Word length distribution (3.4 Counting Other Things)

```python
>>> len1 = [len(w) for w in text1]
>>> fdistl1 = FreqDist(len1)
>>> print(fdistl1)
<FreqDist with 19 samples and 260819 outcomes>
>>> fdistl1.tabulate()

<table>
<thead>
<tr>
<th>3</th>
<th>1</th>
<th>4</th>
<th>2</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>50223</td>
<td>47933</td>
<td>42345</td>
<td>38513</td>
<td>26597</td>
<td>17111</td>
<td>14399</td>
<td>9966</td>
<td>6428</td>
<td>3528</td>
<td>1873</td>
<td>1053</td>
<td>567</td>
<td>177</td>
<td>70</td>
<td>22</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
nltk book: Language Processing and Python

```
fdist1.plot()
```

```
fdist1.plot(cumulative=True)
```