Regular Expressions

Describe the class of strings matched by the following regular expressions:

1. \[a-zA-Z\]+  
2. \[A-Z\][a-z]*  
3. \d+(\d+)?  
4. (\[bcdfghjklmnpqrstvwxyz\][aeiou]\[bcdfghjklmnpqrstvwxyz\])*  
5. \w+|[\^\w's]+  

Write regular expressions to match the following classes of strings:

1. A single determiner (assume that “a”, ”an”, and “the” are the only determiners).
2. An arithmetic expression using integers, addition, and multiplication, such as 2 * 3 + 8.

T9

Write regular expressions that will recognize letters associated to keys on a phone keyboard, i.e.

```
1 2 ABC 3 DEF
4 GHI 5 JKL 6 MNO
7 PQRS 8 TUV 9 WXYZ
```

Write a function, which, given a collection (for example the NPS chat collection:  
http://nltk.googlecode.com/svn/trunk/nltk_data/packages/corpora/nps_chat.zip, for which you can find the cleaned up list of words here: 
http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/magistraleinformatica/eln/nps_chat.zip), collects probabilities from word occurrences, and given a sequence of numbers, displays the most likely words corresponding to those keys, with associated probability.

Zipf’s Law

Let \( f(w) \) be the frequency of a word \( w \) in free text. Suppose that all the words of a text are ranked according to their frequency, with the most frequent word first. Zipf's law states that the frequency of a word type is inversely proportional to its rank (i.e. \( f^r = k \), for some constant \( k \)). For example, the 50th most common word type should occur three times as frequently as the 150th most common word type. (See Foundations of Statistical Natural Language Processing (Manning & Schutze), pp. 23-24, for more information on Zipf's Law.)

Write a Python function \texttt{p4()} to process a large text and plot word frequency against word rank using the nltk.draw.plot graph module (alternatively one can use http://matplotlib.org/). Do you confirm Zipf's law?