SOFTWARE ENGINEER CODING CHALLENGE
BRIGHTEDGE
MANIDEEP ILLENDULA

How to execute:

python ManideepAssignment.py [url]
python ManideepAssignment.py “http://blog.rei.com/camp/how-to-introduce-your-indoorsy-friend-to-the-outdoors/“

Libraries Used:

BeautifulSoup : import bs4
NLTK : import nltk
Urllib2 : import urllib2

Tests:

Input:
http://blog.rei.com/camp/how-to-introduce-your-indoorsy-friend-to-the-outdoors/
Output:
['favorite people', 'favorite activities', 'not-so-outdoorsy friends', 'several breaks', 'far-flung locales', 'good friends', 'flush toilets', 'tasty snacks', 'great outdoors', 'brutal inclines']

Input:
http://www.cnn.com/2013/06/10/politics/edward-snowden-profile/
Output:
['secretive computers', 'harmful effects', 'full rosters', 'major U.S.', 'intelligence-gathering leaks', 'glad Snowden', 'terrorist attacks', 'basic liberties', 'undercover assets', 'overactive Mother']

Input:
http://www.amazon.com/Cuisinart-CPT-122-Compact-2-Slice-Toaster/dp/B009GQ034C/ref=sr_1_1?s=kitchen&ie=UTF8&qid=1431620315&sr=1-1&keywords=toaster
Output:
['2-Slice Toaster', '2-Slice Compact', '12-Cup Programmable', 'wide slots', '4-Year Small', 'similar items', '4-Slice Toaster', 'frozen items', 'new products', 'valid US']

Description:

The program is used to output the top 10 keywords from a given URL. The approach used is as follows

1. Extract the text of the webpage using BeautifulSoup Library.
2. The text which belongs to the parent tag listed is extracted and filtered.
   'article', **'style'**, **'script'**, **'head'**, **'title'**, **'meta'**, **'[document]'**
3. We then tokenize by word and use POS tags to evaluate the keywords and create a map of the keyword with their occurrences.
4. According to my observation a keyword can be a bigram of a noun followed by an adjective or a verb which describes the action of a noun.
5. Hence I designed an algorithm to create a map of all such keywords with the number of occurrences of each keyword.
6. The top 10 keywords are returned as the output.

Functions defined:

1. tagVisible():
   This function extracts the visible text from the webpage using the parent tags mentioned in the description
2. filterText():
   This function filters the extracted text above.
3. analyzeText():
   This function tokenizes the text by word and processes the keywords to form a map of the keywords extracted and the number of occurrences
4. main():
   This function sorts the map in the order of the number of times the keyword occurred and returns the top 10 keywords for the URL as the output.

Future Recommendations:

1. We can use Latent Dirichlet Allocation algorithm to automatically assign categories for the document.
   Ref : http://ai.stanford.edu/~ang/papers/nips01-lda.pdf
2. Stemming and Lemmatization may also improve the results by making the words more similar.
3. N-gram features can also be used when training data is provided which may further improve our results.