INLS 490-172, Capra | Project 1 | Spring 2017
Assigned: Feb 9 | Text Indexing & Boolean Queries

Checkpoint #1 due: 12:00noon, Feb 17
Checkpoint #2 due: 12:00noon, Feb 24
Full project due: 12:00noon, March 3

Overview: Write a Python program to index words from a collection of short stories and then provide an interface to allow users to conduct short Boolean text queries to find stories that match their search.

Learning objectives: Gain experience processing text files, building indexes using Python dictionaries, and using regular expressions.

Project specification:
For this project, you will build an index of words in a collection of short stories. Specifically, we will index the Project Gutenberg text of Grimms' Fairy Tales, by The Brothers Grimm. I have posted a single text file called `grimms.txt` on the course Sakai site that contains the text of all the stories. Use this file for the project. You do not need to index text from any other Project Gutenberg text. When downloading text files from Sakai, you should use your browser's "Save as" feature to save the file exactly as it is posted rather than trying to cut and paste the text into a new file.

The `grimms.txt` file contains short stories (fairy tales) in a single text file. When processing the file, you should **skip over** the introductory lines of text that has information about Project Gutenberg and the table of contents. The short stories start around line 124. Each story starts with a blank line followed by the title of the story in all capital letters on a line by itself, followed by another blank line. After this the text of the story begins. One of the stories, "The Adventures of Chanticleer and the Partlet," has two parts. Treat both parts as belonging to one story. I recommend detecting the story titles from the lines after line 124 rather than using the "CONTENTS" on lines 47-115. The story title lines will be in all capital letters, and some titles may have characters such as a dash (-).

Checkpoint #1: Building the Index – DUE 12:00noon, Feb 17
Your program should read the `grimms.txt` file, and create an index of all the words that appear in the fairy tales except for those in a provided *stopword* list. For this assignment, when building the index, you should REMOVE all characters that are not a letter, number, or space character (e.g., `[a-zA-Z0-9 ]`). In addition, when building the index, you should convert all words to lower-case. You will need to make the same conversions to queries that are entered so that they will match words as you have indexed them.

I have posted a file named `stopwords.txt` on Sakai that contains a list of words[1] (one per line) that should NOT be included in your index. As you are building your index, if you encounter a word in the list of the stopwords, you should skip it.

Do NOT index words that appear outside the text of the fairy tales (e.g., in the Project Guttenberg sections that appear at the beginning and end of the file, and the table of contents). For each word that you include in your index, you should keep track of the stories and line numbers that contain the word. For example, the word "raven" appears in three stories, on the lines show below. Note that the line numbers refer to line numbers in the `grimms.txt` file, starting from line 1 at the top of the file. Hint: one method would be to build a dictionary of dictionaries of lists. The outer dictionary keys are the words, the inner dictionary keys are the story titles, and the list is a list of the line numbers where that word appears in that story.

```
THE LITTLE PEASANT : [3894 , 3924 , 3933 , 3936 , 3939 ]
THE RAVEN : [6765 , 6767 , 6772 , 6773 , 6785 , 6802 , 6807 , 6820 , 6823 , 6839]
SNOWDROP : [4501]
```

[1] The stoplist we are using is adapted from: http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop, filtered to remove words with >=5 chars and with apostrophes.

In my code, I called the outer dict “w2s” (words to stories). Below is an illustration of its structure:

```
>>> w2s['raven']
{'SNOWDROP': [4501], 'THE RAVEN': [6765, 6767, 6772, 6773, 6785,
6802, 6807, 6820, 6823, 6839], 'THE LITTLE PEASANT': [3894, 3924,
3933, 3936, 3939]}
```

For Part 1, you should turn in code that will read the `grimms.txt` file and builds an index using a data structure that allows you to display a list of the story titles and line numbers that contain that word. The `w2s` structure shown above meets these requirements. There are other acceptable ways to construct an index that meets the requirements, but the dict of dict of list approach used in the `w2s` structure is provided as reference for you to use if you wish.

Checkpoint #2: Simple Search Interface – Due 12:00noon, Feb 24
Final Version: Full Search Interface – Due 12:00noon, March 3

After building the index, your program should provide a text-based search interface that that: 1) prompts the user to enter a Boolean search query, 2) returns a list of stories (and other information as outlined below) that match the query, and 3) goes back to (1) to let the user enter a new search query. If the user searches for the string “qquit” (with two qs), then the program should exit. You can use the Python `input()` command to get input from the user.

Boolean query syntax

Your program should support the following types of queries. Note that the first two types of queries are due as part of Checkpoint #2 and the last three are due as part of the final version of the project.

word1	return stories the contain *word1*	Checkpoint #2
word1 or *word2*	return stories that contain either *word1* or *word2*	Checkpoint #2
word1 word2	return stories that contain both *word1* and *word2*	Final version
word1 word2 .. wordn	return stories that contain all the words *word1* through *wordn*	Final version
word1 and *word2*	return stories that contain both *word1* and *word2*	Final version

Output

In response to a query, your program should return information about the query and results as shown below. First, print "query = " and the query string. Then, for each story that matches the query, print the title of the story and a list of all the lines that contain the query string. Use indentation to improve readability as shown below. You should also highlight the word that was matched on the line by surrounding it with two asterisks and displaying the word in all capital letters.

```
Please enter your query: owl

query =  owl
     SNOWDROP
       4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
     JORINDA AND JORINDEL
       542 an **OWL**, or crept about the country like a cat; but at night she always
       581 with a mournful _jug, jug_. An **OWL** with fiery eyes flew three times
       588 the gloomy night came; the **OWL** flew into a bush; and a moment after the
```

For queries that have no matching stories, you should print "--" as shown below.

```
Please enter your query: python

query =  python
    --
```

For queries that involve multiple words, for each story that matches the query, you should print results for each word as shown below. Use indentation to improve readability as shown below.

```
Please enter your query: owl raven

query =  owl raven
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       raven
         4501 an owl, and then a **RAVEN**, and at last a dove, and sat by her side.
```

For words in a an “or” query that do not occur in a story, you should print “--" as shown below.

```
query =  owl or raven
     <<some lines omitted for space>>
     JORINDA AND JORINDEL
       owl
         542 an **OWL**, or crept about the country like a cat; but at night she always
         581 with a mournful _jug, jug_. An **OWL** with fiery eyes flew three times
         588 the gloomy night came; the **OWL** flew into a bush; and a moment after the
       raven
        --
```

I have included a larger set of example queries and output at the end of this assignment document.

Advanced queries

If you implement the program as outlined so far, you can earn up to a maximum of 92 out of 100 points. You can earn additional points for implementing two additional features:

1. (4 points) Add a query keyword “morethan” that will allow searches that require the word to the left to appear more than a specified number of times. On the right, the user should be able to specify either another word, or a number. For example, “owl morethan 5” should return stories in which the word owl appears more than 5 times, and “owl morethan raven” should return stories in which the word owl appears more times than the word raven.

2. (4 points) Add a query keyword “near” that will allow users to enter queries that will search for two words that occur within plus or minus 1 line of each other in a story. For example, “owl near raven” would match stories in which the word owl appeared either on the same line as raven, or one line above or below it.

Grading:

Your program will be evaluated based on its functionality, programming logic, and programming style. Functionality focuses on the question, “Does your program product the correct results?” Programming logic considers whether the approach you implemented in your code is correct (or close to correct). Programming style looks at how easy it is to understand your code – is it organized well, did you use functions appropriately, did you include good comments?

How to turn in your assignment:

Your program should be contained in a single file and **be entirely code that you wrote yourself**. Name your file according to the following convention:

```
youronyen_p1.py
```

Replace *youronyen* with your actual Onyen (e.g. my assignment would be `rcapra_p1.py`).

I will test your program by running it with Python 3.6, with the files `grimms.txt` and `stopwords.txt` in the same directory.

Submit your file electronically through the Sakai by going to the Assignments area and finding the Project 1 assignment. After you think you have submitted the assignment, I recommend checking to be sure the file was uploaded correctly by clicking on it from within Sakai. Keep in mind that if I cannot access your file, I cannot grade it.

If for some reason you need to re-submit your file, you must add a version number to your filename (e.g., `youronyen_p1_v2.py`).

Sakai is also configured with a due date and an "accept until" date. Unless you have made arrangements with the instructor in advance, submissions received after the due date may receive a 5% penalty per day.

Example Program Run

```
Loading stopwords...
['a', 'able', 'all', 'also', 'am', 'an', 'and', 'any', 'are', 'as', 'ask', 'at', 'away', 'b', 'be',
'been', 'best', 'both', 'but', 'by', 'c', 'came', 'can', 'cant', 'co', 'com', 'come', 'd', 'did',
'do', 'does', 'done', 'down', 'e', 'each', 'edu', 'eg', 'else', 'et', 'etc', 'even', 'ever', 'ex',
'f', 'far', 'few', 'five', 'for', 'four', 'from', 'g', 'get', 'gets', 'go', 'goes', 'gone', 'got',
'h', 'had', 'has', 'have', 'he', 'help', 'her', 'here', 'hers', 'hi', 'him', 'his', 'how', 'i', 'ie',
'if', 'in', 'inc', 'into', 'is', 'it', 'its', 'j', 'just', 'k', 'keep', 'kept', 'know', 'l', 'last',
'less', 'lest', 'let', 'like', 'look', 'ltd', 'm', 'many', 'may', 'me', 'mean', 'more', 'most',
'much', 'must', 'my', 'n', 'name', 'nd', 'near', 'need', 'new', 'next', 'nine', 'no', 'non', 'none',
'nor', 'not', 'now', 'o', 'of', 'off', 'oh', 'ok', 'okay', 'old', 'on', 'once', 'one', 'ones', 'only',
'onto', 'or', 'our', 'ours', 'out', 'over', 'own', 'p', 'per', 'plus', 'q', 'que', 'qv', 'r', 'rd',
're', 's', 'said', 'same', 'saw', 'say', 'says', 'see', 'seem', 'seen', 'self', 'sent', 'she', 'six',
'so', 'some', 'soon', 'sub', 'such', 'sup', 'sure', 't', 'take', 'tell', 'th', 'than', 'that', 'the',
'them', 'then', 'they', 'this', 'thru', 'thus', 'to', 'too', 'took', 'try', 'two', 'u', 'un', 'unto',
'up', 'upon', 'us', 'use', 'used', 'uses', 'uucp', 'v', 'very', 'via', 'viz', 'vs', 'w', 'want',
'was', 'way', 'we', 'well', 'went', 'were', 'what', 'when', 'who', 'whom', 'why', 'will', 'wish',
'with', 'x', 'y', 'yes', 'yet', 'you', 'your', 'z', 'zero']

Building index...
1 THE GOLDEN BIRD
2 HANS IN LUCK
3 JORINDA AND JORINDEL
4 THE TRAVELLING MUSICIANS
5 OLD SULTAN
6 THE STRAW, THE COAL, AND THE BEAN
7 BRIAR ROSE
8 THE DOG AND THE SPARROW
9 THE TWELVE DANCING PRINCESSES
10 THE FISHERMAN AND HIS WIFE
11 THE WILLOW-WREN AND THE BEAR
12 THE FROG-PRINCE
13 CAT AND MOUSE IN PARTNERSHIP
14 THE GOOSE-GIRL
15 THE ADVENTURES OF CHANTICLEER AND PARTLET
16 RAPUNZEL
17 FUNDEVOGEL
18 THE VALIANT LITTLE TAILOR
19 HANSEL AND GRETEL
20 THE MOUSE, THE BIRD, AND THE SAUSAGE
21 MOTHER HOLLE
22 LITTLE RED-CAP [LITTLE RED RIDING HOOD]
23 THE ROBBER BRIDEGROOM
24 TOM THUMB
25 RUMPELSTILTSKIN
26 CLEVER GRETEL
27 THE OLD MAN AND HIS GRANDSON
28 THE LITTLE PEASANT
29 FREDERICK AND CATHERINE
30 SWEETHEART ROLAND
31 SNOWDROP
32 THE PINK
33 CLEVER ELSIE
34 THE MISER IN THE BUSH
35 ASHPUTTEL
36 THE WHITE SNAKE
37 THE WOLF AND THE SEVEN LITTLE KIDS
38 THE QUEEN BEE
39 THE ELVES AND THE SHOEMAKER
40 THE JUNIPER-TREE
41 THE TURNIP
42 CLEVER HANS
43 THE THREE LANGUAGES
44 THE FOX AND THE CAT
45 THE FOUR CLEVER BROTHERS
46 LILY AND THE LION
47 THE FOX AND THE HORSE
48 THE BLUE LIGHT
49 THE RAVEN
50 THE GOLDEN GOOSE
51 THE WATER OF LIFE
52 THE TWELVE HUNTSMEN
53 THE KING OF THE GOLDEN MOUNTAIN
54 DOCTOR KNOWALL
55 THE SEVEN RAVENS
56 THE WEDDING OF MRS FOX
57 FIRST STORY
58 SECOND STORY
59 THE SALAD
```

```
60 THE STORY OF THE YOUTH WHO WENT FORTH TO LEARN WHAT FEAR WAS
61 KING GRISLY-BEARD
62 IRON HANS
63 CAT-SKIN
64 SNOW-WHITE AND ROSE-RED


Welcome to the Grimms' Fairy Tales search system!


Please enter your query: owl

query =  owl
     SNOWDROP
       4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
     JORINDA AND JORINDEL
       542 an **OWL**, or crept about the country like a cat; but at night she always
       581 with a mournful _jug, jug_. An **OWL** with fiery eyes flew three times
       588 the gloomy night came; the **OWL** flew into a bush; and a moment after the

Please enter your query: owl raven

query =  owl raven
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       raven
         4501 an owl, and then a **RAVEN**, and at last a dove, and sat by her side.

Please enter your query: owl and raven

query =  owl and raven
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       raven
         4501 an owl, and then a **RAVEN**, and at last a dove, and sat by her side.

Please enter your query: owl or raven

query =  owl or raven
     THE LITTLE PEASANT
       owl
        --
       raven
         3894 the way he passed by a mill, and there sat a **RAVEN** with broken wings,
         3924 in which the **RAVEN** was, lying on the ground, and asked: 'What have you
         3933 and found the wine. 'Now go on,' said he. The peasant made the **RAVEN**
         3936 went thither, and found the roast meat. The peasant made the **RAVEN**
         3939 went there and found the salad. At last the peasant pinched the **RAVEN**
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       raven
         4501 an owl, and then a **RAVEN**, and at last a dove, and sat by her side.
     JORINDA AND JORINDEL
       owl
         542 an **OWL**, or crept about the country like a cat; but at night she always
         581 with a mournful _jug, jug_. An **OWL** with fiery eyes flew three times
         588 the gloomy night came; the **OWL** flew into a bush; and a moment after the
       raven
        --
     THE RAVEN
       owl
        --
       raven
         6765 were a **RAVEN** and would fly away, then I should have a little peace.'
         6767 turned into a **RAVEN**, and flew away from her through the open window. The
         6772 a **RAVEN** calling, and he followed the sound of the voice. As he drew
         6773 near, the **RAVEN** said, 'I am by birth a king's daughter, but am now under
         6785 The man promised to do all that she wished, but the **RAVEN** said, 'Alas! I
         6802 and mounted the tan-heap to await the **RAVEN**. Suddenly a feeling of
         6807 o'clock the **RAVEN** came driving along, drawn by her four white horses;
         6820 watch for the **RAVEN**. He had not been there long before he began to feel
         6823 As the **RAVEN** drove along her four chestnut horses, she said sorrowfully
         6839 he slept like a log. At two o'clock the **RAVEN** could be seen approaching,
Please enter your query: owl dove raven

query =  owl dove raven
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       dove
         4501 an owl, and then a raven, and at last a **DOVE**, and sat by her side.
       raven
         4501 an owl, and then a **RAVEN**, and at last a dove, and sat by her side.

Please enter your query: python

query =  python
    --

Please enter your query: owl or python

query =  owl or python
     SNOWDROP
       owl
         4501 an **OWL**, and then a raven, and at last a dove, and sat by her side.
       python
        --
     JORINDA AND JORINDEL
       owl
         542 an **OWL**, or crept about the country like a cat; but at night she always
         581 with a mournful _jug, jug_. An **OWL** with fiery eyes flew three times
         588 the gloomy night came; the **OWL** flew into a bush; and a moment after the
       python
        --

Please enter your query: owl and python

query =  owl and python
    --

Please enter your query: owl dove python

query =  owl dove python

Please enter your query: the

query =  the
    --

Please enter your query: able

query =  able
    --

Please enter your query: best

query =  best
    --

Please enter your query: quit

query =  quit
    --

Please enter your query: qquit
>>>
```