CSC498 – Introduction to Natural Language Processing
Homework 1: Text Preprocessing – Regular Expressions
Due date: Friday October 09 - Midnight
Submission: Blackboard upload.

1. Regular Expressions. Using Python and the ***re*** package, write regular expressions for the following:
    a. The set of all strings from the alphabet {a, b} such that each ***a*** is immediately preceded by and immediately followed by a ***b***.
    b. Write a Python program to remove lowercase substrings from a given string. Test it on string: **"KDeoBNBklvVHuipfLLaeioU".**
    c. Match a string of any length and any character.
    d. The set of consecutive repeated words, such as "ciao ciao sigñorina" or "so very very far away":
        i. After matching consecutive repeated words, remove the duplicate word (i.e. occurring second). To this end, you may use the ***substitution*** function of the ***re*** package. The function's prototype is:
            1. **re.sub(pattern, repl, string)** where *pattern* is the string to replace, *repl* is the replacement string, and *string* is the sample sentence you're searching in.
    e. To search the numbers (0-9) of length between 1 to 3 in a given string. Test it on the string: "Busy people fact: 2, 8, 27, 34, 090, 828. You were too busy to read that number."
    f. The **re.compile(pattern, repl, string)** converts a string pattern into a RegexObject, which separates the definition of the regex from its use. This is useful for pattern matching, as you may use the object to search for the same pattern again without having to rewrite it.
        i. Repeat question (d) using **re.compile**.

    g. Write a Python regex that matches a word at the end of a string, with optional punctuation. Test it on the following two sample sequences:
    "If one is lucky, a solitary fantasy can totally transform one million realities."
    "If one is lucky, a solitary fantasy can totally transform one million realities. "
    h. Write a function to convert a date format from: **yyyy-mm-dd** to **dd-mm-yyyy**.
    i. Write a regex to remove words in a string, having length between 1 and 3. Test it on the following sentence: "The black horse neighs to the fearful fox."
    j. Write a regex to remove all whitespaces from a string.
    k. Compare the similarity of the two verbs: "remain" and "prevail".
        i. Provide a definition to each.
        ii. Measure the similarity between the two.

2. Text Preprocessing. In this part, you will implement a Python program that preprocesses a given text by applying normalization techniques, in order to make the text consistent before any NLP operation is performed on it. The normalization step consists of the following subtasks:
    a. Load a text file of your choice from the Gutenberg corpus.
    b. Print a sample of the raw text.
    c. Convert the entire text to lowercase
    d. Remove all the numbers, if any.
    e. Remove punctuations from the text, such as ".?!", but also remove characters like "@$#".
    f. Show the size of the vocabulary, i.e. the number of distinct words.
    g. Compute the lexical diversity of the text.
    h. Plot the top 30 most frequently used words.
    i. Implement sentence tokenization.
    j. Implement word tokenization.
    k. Remove stop words.
    l. Lemmatize the '*verbs*' only in the text.
    m. pyspellchecker is a Python package that allows you to perform spelling corrections, as well as see candidate spellings for a misspelled word. Install pyspellchecker on your computer, and then import it:
        i. `from spellchecker import SpellChecker`
           `spell = SpellChecker()`
        ii. Load any sentence from your text into string **checksent**, and using function **correction(str)**, check if there are any misspelled words.
           Hint. Test the function on any sentence such as:
           "Do you percieve that this sentnce has mispelled werds".