Homework 1

Exercise 1 (regex)

What do these regular expressions match? What the eventual groups defined in them return? Describe it in your own words, and with some examples of matching strings.

1. `'[A-Z][a-z]+'`
2. `'[+-]?[0-9]+\.?[0-9]*'`
3. `'(https?)://([\w.]*\w+\.(\w+))(/?[\S]*)'`
4. `"([A-Za-z]+)n't"` note the double quotes around the string, so that the ' doesn't need to be escaped with a backslash
5. `'([A-Z]{3})([A-Z]{3})([0-9]{2})([A-Z])([0-9]{2})([A-Z][0-9]{3})([A-Z])'`

An *hashtag* is a sequence of characters starting with a '#' (i.e, it is not preceded by a letter, a number or an underscore "_") and it is followed by letters (at least one, in any position), numbers, and underscores.

For example, this string contains six hashtags:

```
'#aaa #eee,#122a -#AbC_d #__dde #1__d'
```

while this string contains no hashtags, despite containing some #:

```
'aa#eee #122 _#AbC_d #___ #1___'
```

6. write a function **get_hashtags** that takes in input a string and return a list of all the hashtags that appear in the input string, e.g.:

```
>get_hashtags('#aaa #eee,#122a -#AbC_d #__dde #1__d')
Out: ['aaa', 'eee', '122a', 'AbC_d' ,'__dde', '1__d']
>get_hashtags('aa#eee #122 _#AbC_d #___ #1___')
Out: []
```

Hint: have a look at re.findall

Exercise 2 (word distribution)

Get the file at https://www.gutenberg.org/files/52484/52484-0.txt.

1. Open it, read it, convert it to lowercase, split the text into a sequence of words (let's define a word as a sequence of letters, discard all the rest, hint: re.split).

2. Using the sequence of words produced in the previous step, build a vocabulary of all the distinct words and count the frequency of each word.
3. Print the 30 most frequent words, starting from the most frequent one.
4. Write a function **at_least** that returns the number of words appearing at least n times in the text.

   ```
   >at_least(1)
   Out: 6000 # this is a fake number, it should equal to the number of distinct words in text
   ```

5. Compute all the values of at_least function for n that goes from 1 to the highest frequency. Save all such `(n, at_least(n))` pairs in a csv file.
6. Plot the sequence of pairs in a log-log plot and observe if the distribution of values follows a Zipf's law (you can import your csv data in a spreadsheet to make the plot or try to use a python plotting library, such as matplotlib).

Exercise 3 (next word probability)

1. Using the sequence of words produced in the step 1 of exercise 2, write a function **next_word** that given a word returns a list of the possible next words, as observed in text, giving for each word the relative probability.

   For example, if the word 'bella' appears in text three times, one time followed by 'fortuna' and the other two times followed by 'giornata', the output of next_word is:

   ```
   >next_word('bella')
   Out: [('fortuna',0.33333), ('giornata',0.66666)]
   ```

   Return an empty list when the word is not in the vocabulary.

   ```
   >next_word('abcdef')
   Out: []
   ```

2. Use the next_word function to generate some text, starting from a seed word

   ```
   >generate_text('bella', 20)
   bella cosa volete che imparerò a casa dalla parte con un vento impetuoso di legno galleggiava facilmente e chi lo chiamavano
   ```