# Programming Assignment 2 – Basic Probability

2021 – 2022, Master of Logic, University of Amsterdam

Written by: Jakub Dotlačil

## 1 Exercise

This week we will start to work with some real data. This website Project Gutenberg[1] provides access to thousands of books which have run out of copyright. They are freely downloadable for you to read. Our goal is to analyse these books for some of their basic properties. In particular, we want to find their most frequent and least frequent words and also find the counts for specific words. For example, we might want to know how often the word *Alice* occurs in Alice in Wonderland.

Our program will be able to read text files. During development you can experiment with any book you like. However, when grading we will all use this version of War and Peace.[2]

As you will notice, the file contains some document information from Project Gutenberg. This means that our word counts will not be entirely correct as there are also words that do not belong to the main text. We will not be bothered by this, however.

One last remark: when you download any book of your choice, please make sure to download the `Plain Text UTF-8` version of it. Otherwise your program will not be able to read the file (at least not without some additional effort).

Here is what your program must be able to do:

- Ask the user for a path to a text file and read that text file into memory (we will help you with this, part of the code for this are already present)
- Run four functions:
  1. The first function, called collect_frequencies, will take the name of the file provided from the user, and it returns a dictionary with frequency counts for every word in the file
  2. The second function, called find_frequent_words, has an optional argument, amount, which specifies how many frequent words should be found. It outputs two lists. The first list consists of the most frequent words, as many as specified in amount. The second list is the list of the

[1] https://www.gutenberg.org/

[2] https://www.gutenberg.org/cache/epub/2600/pg2600.txt

frequencies of the corresponding words. The default value of amount is 50.

3. The third function, called find_word, has an optional argument, word. It returns the frequency of the word if it is present in the file. If the word is not present, the function returns nothing (and prints a message that the word was not found). If no value for the argument word is specified, the function randomly selects a word from the file.
4. The fourth function, print_lists, which prints the output of find_frequent_words and find_word.

Notice that reading the text anew for every operation would be extremely wasteful. You should try to get your program to memorize all relevant information.

## 2 Grading

The task is to count words regardless of casing. This means that $The$ and $the$ should be treated as the same word. To achieve this, all words should be lowercased before counting them. However, we are not going to worry about punctuation. For us $Alice$ and $Alice?$ are different words. We define a word as any sequence of characters that is divided from other words by white spaces.

1 point When the program starts it asks the user for a path to a file (including the file name). It is ok for the program to crash if this path does not exist.

2 points The program reads the text file only once and stores all relevant information in a dictionary. The functions find_frequent_words, find_word do not operate on the file, only on the dictionary created from the file. Depending on your computer, reading a file can take several seconds.

1 point The program generates a dictionary using the function collect_frequencies, with words as keys and integers as values.

2 points The program prints the number $n$ of most frequent words to the screen as a list in which each line looks as follows (1 point):

$$word \quad wordCount$$

If there are several words of the same count, they are ordered alphabetically (1 point).

2 points The count of specific words is printed in the same format as the above two items (1 point). If a word does not occur in the text, the program prints *Sorry, this word does not occur in the text* (1 point).

1 point The function find_frequent_words has an optional argument, which has the default value of 50 (that is, by default, 50 most frequent words are returned). The default value can be changed and the function still works correctly.

1 point The function find_word has an optional argument, word. If no value is specified for this argument, the function selects a word from the file at random. If the value is specified, the function correctly returns only the information about that word.

## 3 Counters

For this exercise, it may be helpful to use a Python data structure known as Counter. It is not strictly necessary to use counters for this exercise but they might make your life a lot easier. Also make sure to the check the official documentation.