# Advanced Python Course - Assignment 2

October 15, 2024

Advanced Python Course

Assignment 2

Each task is worth 2 points. You must present three tasks for evaluation in the lab.

Task 1.

In general elections for parliament or local government, seats are distributed between electoral committees using the d'Hondt method. Write a function that takes as arguments the election results in the form of the number of votes cast for each committee and the number of seats to be filled. The result should be the number of seats allocated to each committee.

We assume that:

- The details of the d'Hondt method are as described in https://pl.wikipedia.org/wiki/Metoda_D%E2%80%99Hondta;
- The election threshold is 5% (we do not consider minority or ethnic electoral committees, which are exempt from the election threshold).

Task 2.

Write a Python function sudan(n, x, y) that calculates the following recursive function:

- $F_0(x, y) = x + y$
- $F_{n+1}(x, 0) = x$, for $x >= 0$

- Fn+1(x, y + 1) = Fn(Fn+1(x, y), Fn+1(x, y) + y + 1)

Since this function grows very rapidly, be careful not to test for n > 2. To speed up the function, implement memoization (caching of already computed results). Experimentally determine the largest arguments for which the function is reasonable to call without memoization and with memoization. Include the results in a comment in the source file.

Task 3.

Each natural language has its own characteristic frequency of letter occurrences. Write two functions: one that calculates statistics for at least three languages based on literary works, and another that determines the most likely language in which a given text is written based on these statistics.

You can assume that only languages that use an alphabet derived from the Latin alphabet are considered. The texts should be stored in plain text files. A good source for texts in various languages is Project Gutenberg or Wikisource.

Task 4.

Overly complicated sentences can be troublesome for the reader. We will simplify sentences in the following way:

- First, remove excessively long words;
- Then, randomly remove words if the sentence contains too many.

Write a function simplify_sentence(text, word_length, word_count), where word_length is the maximum allowed word length, and word_count is the maximum number of words that can be in the sentence. For example:

text = "The periclinal division of the spindle-shaped initials of the cambium is characterized by a division wall initiated in the maximum plane."

The result of simplify_sentence(text, 10, 5) should be something like:

"The cambium is characterized by the division."

Test your program on a popular literary work that is legally available online. Include in the source file either the code that retrieves such text or a link to the text.

Task 5.

One of the simplest methods of text compression is to replace a sequence of identical characters with a pair (character, count). For example, instead of 'aaaaaa', we can use [(5, 'a')], and we write a single letter as it is. For example, 'suuuuper' would be compressed to [(1, 's'), (4, 'u'), (1, 'p'), (1, 'e'), (1, 'r')].

Write two functions: compress(text) and decompress(compressed_text), which return the compressed text (i.e., a list of tuples) and the decompressed text, respectively. You can assume that only texts containing letters and punctuation marks are compressed.

Test your program on a longer text that is legally available online. Include a link to the text in your code or include the code that retrieves this text and calls this function.

Marcin Mlotkowski