Python for data scientists SP1 2021/2022

Assignment 2

There are two problems in this assignment.

You can either submit:

- Two Python files for Problem 1 (solutionA2_P1_1.py for the simple solution and solutionA2_P1_2.py for the efficient solution)
- One Python file for Problem 2 (solutionA2_P2.py)
- A PDF file for the rest of the questions and for explanation part.

Or a Jupyter Notebook with clearly marked solutions for which problems and which tasks.

Note:

- If you add any details or make any ssumptions, please clearly describe in your submission.

Problem 1

Looking for a word in a dictionary is very easy since words are sorted alphabetically. But, what if we are looking for all the words for which the second letter is an *a,* it's impossible unless we sort all the words by their second letter. In this case, we would need a different dictionary for each letter position.

The objective of this exercise is to design and implement a kind of dictionary that will allow you to find all the words with a given letter in a given position.

Tasks:

1. Write the simplest solution that allow you to find all the words with a given letter in a given position.
2. Write a more efficient solution by using a different data structure. Explain and motivate your choice.
3. Compare the execution time between these two solutions.

Examples are shown below. For this, let's assume that:

- **d** is the data structure you will use for this problem;
- **words** represent a sample of words from a dictionary. You can also use the file *words.txt*.

words = ['class', 'case', 'course', 'dictionary', 'java', 'list', 'program', 'python', 'tuple', 'word']

---

w = words_letter_position(d, 'a', 1)
print("results = ", w)

output:
results = ['case', 'java']

---

w = words_letter_position(d, 't', 3)
print("results = ", w)

output:
results = ['dictionary', 'list']

Problem 2

An anagram is a word formed by rearranging the letters of a different word by using all the original letters exactly once. For example, the word **teacher** can be rearranged into **cheater** or the word **rebuild** can be rearranged into **builder**.

Tasks:

1. Write a program that reads a word list from the file *words.txt* and as an output it prints to a destination file (the file name is given as an input):

- all the lists of words that are anagrams ordered by the length of lists.

- all the lists of words that are anagrams and contains n letters (n is given as an input).

Note that all anagrams should also be in the *words.txt* file.

An example:

For 6 letters, we will have for example:

...

(2, ['append', 'napped'])

...

(12, ['enters', 'ernest', 'estren', 'nester', 'renest', 'rentes', 'resent', 'sterne', 'streen', 'tenser', 'ternes', 'treens'])

...

2. Which data structure did you use and why?