Python for Engineers 0509-1820, Semester 1 2019-2020

Homework 4

General instructions

- Read the questions **carefully** and make sure your programs work according to the requirements.
- The homework needs to be done individually!
- Read the submission rules on the course web page. All the questions need to be submitted together in the file ex1_012345678.py attached to the homework, after changing the number 012345678 with your ID number (9 digits, including check digit).
- How to write the solution: in this homework, you need to complete the code in the attached outline file.
- Check your code: in order to ensure correctness of your programs and their robustness in the presence of faulty input, for each question run your program with a variety of different inputs, those that are given as examples in the question and additional ones of your choice (check that the output is correct and that the program does not crash).
- Some of the questions are checked automatically. You thus need to write your solutions only in the specified spaces in the outline file.
- Unless stated otherwise, you can suppose that the input received by the functions is correct.
- You are not allowed to change the names of the functions and variables that already appear in the attached outline file.
- You are not allowed to erase the instructions that appear in the outline file.
- You are not allowed to use outside libraries (you may not use **import**)
- Final submission date: see course web page.

Dictionaries

Question 1

Write a function called *second_most_popular_character* which receives a string and returns the second most popular letter in this string. In case there is more than one such letter, you need to return the smallest among them (in the alphabetic order).

Example:

```
>>> second_most_popular_character('HelloWorld')

'o'
```

Explanation: the letter ‘l’ appears three times and the letter ‘o’ twice. The other letters appear only once. Hence ‘o’ is the second most popular letter.

```
>>> second_most_popular_character('cccaabb')

'a'
```

Explanation: letters ‘a’ and ‘b’ appear twice, and ‘c’ appears three times. We return ‘a’ because it is smaller than ‘b’.

Instructions: in order to solve this problem, you need to follow the following steps:

1. Build a dictionary that contains the letters of the string and the number of occurrences of each letter.
2. Find the second largest value in the dictionary.
3. Find the smallest key among these values (in case there are several)

You can assume that the string contains only letters.

Clarifications regarding question 1:

a) You can assume that the function receives a string that contains more than one letter. That is, the parameter cannot be ‘a’ or ‘aaa’.

b) Lowercase letters are considered different than uppercase letters. That is, ‘a’ and ‘A’ are different letters.

c) In Python, uppercase letters are smaller than lowercase letters. That is A-Z < a-z. For example: ‘A’<’a’ and ‘Z’<’z’.

d) In case there are several different letters that have the highest number of occurrences, for example ‘aaabbbcc’, you need to return one of the following two options:

    a. The letter ‘c’ because it occurs twice whereas ‘a’ and ‘b’ occur three times.

    b. The letter ‘a’ because it appears with the highest number of occurrences and the second highest number of occurrences (like ‘b’) and it is the smallest letter among them.

    Both solutions will be regarded as valid by the graders.

Question 2

In the recitation, we saw an implementation of a sparse matrix using a dictionary. In this implementation, for every item of the matrix which is not 0, a key of type *tuple* is stored in the dictionary. This tuple represents the coordinates of the item, and the value represents the value of the item in the matrix. Implement the function *diff_sparse_matrices(lst)* which receives a list of dictionaries (two or more) representing sparse matrices and which returns a dictionary the matrix of the difference between the matrices in the list.

Let lst = [M1, M2, …, Mn]

The difference matrix is calculated as a difference of all the elements in their respective indices.

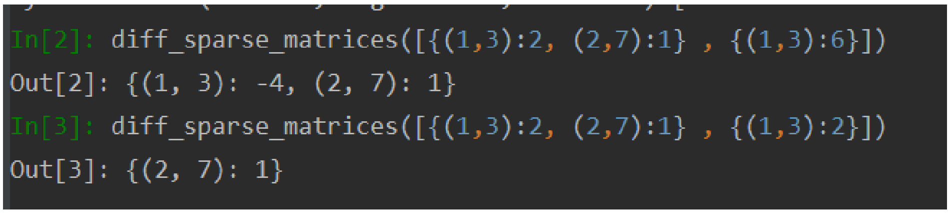
For example:

M1 – M2 = M1(i, j) – M2(i, j), for every i, j.

It can be calculated for more than two matrices, as follows:

M1 – M2 – M3 - … - Mn = M1(i, j) – M2(i, j) – M3(i, j) - … - Mn(i, j) , for every i, j.

For example:



```
In[2]: diff_sparse_matrices([{(1,3):2, (2,7):1} , {(1,3):6}])
Out[2]: {(1, 3): -4, (2, 7): 1}
In[3]: diff_sparse_matrices([{(1,3):2, (2,7):1} , {(1,3):2}])
Out[3]: {(2, 7): 1}
```

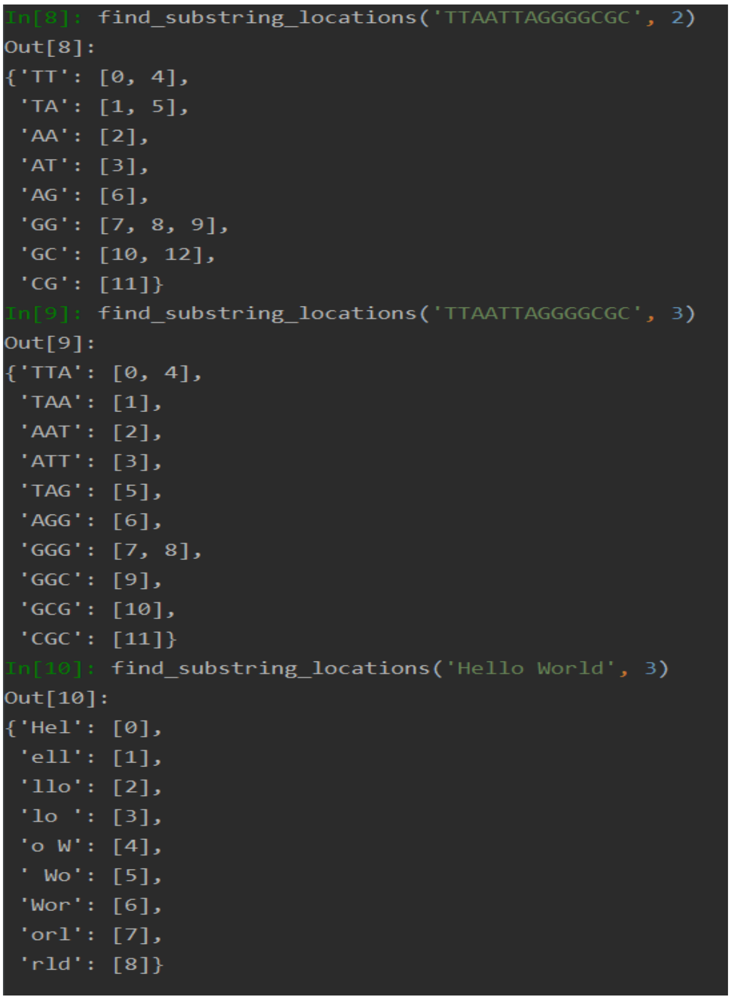
In the second case, the value of the difference matrix at coordinate (1,3) is 0, hence this value is removed from the dictionary. The function is supposed to receive only a list containing matrices of the type described above. The list can contain two or more such matrices. The lengths of the lists are not necessarily equal.

The number of items in each dictionary is not necessarily identical (i.e. the number of items different than 0 not necessarily the same in each matrix).

Question 3

Implement a function called *find_substring_locations(s, k)* which receives a string and the length of a substring as an integer, and which returns the following dictionary:

- The keys are all the substrings of the string *s*, of length *k* (a substring is a continuous sequence of characters inside *s*).
- The value associated with each key is the list of all indexes in which it appears in *s* (every position is indicated by the index of the first character of the substring). For example: “ge” inside “dgre” will appear at position 2.Example:



```
In[8]: find_substring_locations('TTAATTAGGGGCGC', 2)
Out[8]:
{'TT': [0, 4],
 'TA': [1, 5],
 'AA': [2],
 'AT': [3],
 'AG': [6],
 'GG': [7, 8, 9],
 'GC': [10, 12],
 'CG': [11]}
In[9]: find_substring_locations('TTAATTAGGGGCGC', 3)
Out[9]:
{'TTA': [0, 4],
 'TAA': [1],
 'AAT': [2],
 'ATT': [3],
 'TAG': [5],
 'AGG': [6],
 'GGG': [7, 8],
 'GGC': [9],
 'GCG': [10],
 'CGC': [11]}
In[10]: find_substring_locations('Hello World', 3)
Out[10]:
{'Hel': [0],
 'ell': [1],
 'llo': [2],
 'lo ': [3],
 'o W': [4],
 ' Wo': [5],
 'Wor': [6],
 'orl': [7],
 'rld': [8]}
```

As you can see, the function divides the string in substrings of length *k*, in a continuous way, and counts how many times each substring appears in string *s*.

The function needs to be able to handle *k* of size:

1 <= k <= len(s)

I/O and Exceptions

Question 4

In a job interview in a company that specializes in the automated treatment of text, you are asked to implement the function *count_lines(in_file, out_file)* which receives two strings which represent pathnames of files. The function will write to the output file *out_file* the number of lines in the file *in_file*.

In this question you may assume that the input file is a correct text file, and you do not need to take care of error cases.

- Attached to this homework is the file *q4_input_example_1.txt* for testing the function. It is recommended to create additional test files of your own for borderline cases.

Example:

The file *q4_input_example_1.txt* contains the following text:

line 1

line 2

line 3 -> thus, your code should write to the output file 3

Running the function with the input *in_file* which contains the path name to this file, and an additional pathname *out_file*, will write a new file in the given pathname out_file, which contains the following text:

3

Question 5

You have obtained the above-mentioned job, and as your first assignment you are asked to automatically classify the contents of the documents between 'happy' and 'sad'.

You are of course eager to use the most advanced machine learning techniques but the head of the team is reminding you that an important practice in engineering is to begin with a simple solution and to use it to check if a more complicated solution is called for.

The head of the team is thus asking you to develop the function *simple_sent_analysis(in_file)*.

This function receives the pathname of the text file that needs to be classified (a variable called *in_file*) and returns a dictionary that contains the number of times the word "happy" appears, and the number of times that the word "sad" appears, in the following way:

{'happy' : num_happy, 'sad' : num_sad}

You can assume that the text in the input file contains only English letters, numbers, spaces, and the following characters: %$;- ,?! . The file can contain several lines.

- The counting must be case insensitive, i.e. 'happy', 'Happy', 'hapPY' and 'HAPPY' are counted as 'happy'.
- You need to make sure not to count words that *contain* 'happy' or 'sad'. For example, the word 'saddle' should not count as 'sad'.
- You should not consider the characters in the parentheses hereafter: (!?, -;$%.). That is, you need to count 'happy' and 'sad' even if one of these characters occurs before or after the word. For example, '?happy' or 'happy%' do count as 'happy', but 'hap?py' does not count as 'happy'.
- If an *IO* exception occurs, you need to catch it, and to finish the execution in a clean way (without crashing) and to return an empty dictionary. If the file was opened, you need to make sure that the file gets closed. You also need to print the message: '*Cannot encode $in_file due to IO error*', where *$in_file* represents the name of the input file.
- A file called *q5_input_example_1.txt* is attached to this homework for testing the function. It is recommended to create additional files for testing other cases, including borderline cases.
- Using string methods can help you a lot. For example, the method *str.replace(old, new)* enables you to replace each occurrence of the string *old* in the string *str* by the string *new*.
- For those interested in the subject (outside the scope of this class), you can read more here about Sentiment Analysis.

Example:

The file *q5_input_example_1.txt* contains the following text (the colors are for explanation purposes only, they do not appear in the text itself):

,happy hap saddle

sad bla sad s!ad

shimi happy!

Executing your function on this file should return the following dictionary: *{'happy': 2, 'sad': 2}.*

Question 6

The document classification procedure that you implemented above works very well, and thanks to it the company got a job from a TV series production company.

The production company wants to know if there is more profit in producing 'happy', 'sad', or 'neutral' series. The company provided scripts of all their series. The staff of the application has already run your algorithm on all the scripts and gave you a CSV ("comma-separated values") file that contains the name of the series, its profit, and the classification (happy, sad, neutral).

Implement the function *calc_profit_per_group(in_file)* which receives as input the CSV file and returns a dictionary containing the average profit for each category.

In case one of the categories does not appear at all, the average profit will be replaced by 'NA'. It is considered as a valid case, and no exception should thus be raised (see second example below).

- If an exception of type *IO* occurs, you need to catch it, and to end the execution in a clean way (without crashing). You should not write anything to the output file. You should also print the message '*Cannot use $in_file due to IO error.*', where *$in_file* represents the name of the input file.
- If a series appear more than once, an exception of type ValueError will be raised, with the string 'The series *$series_name* appears more than once', where *$series_name* represents the name of the series and nothing will be returned. In case there is more than one series that appears several times, it is enough to provide the name of the first series that appears twice (see third example).
- You need to perform the following checks on the input:
    - ○ Check that there are 3 columns
    - ○ The second column contains only numbers
    - ○ The third column contains only the values sad/happy/neutral in lowercase

In case an error is found, you need to raise a *ValueError* exception with the message 'Invalid input' (see example 4).

A file called *q6_input_example_1.txt* is attached to this homework for testing the function. It is recommended to create additional files for testing other cases, including borderline cases.

Example 1:

Text in *in_file* :

```
Descendant Without A Conscience,505.4,happy

Wolf Of The Solstice,30000,sad

Women Of Hope,-4000,neutral

Pirates Of Perfection,65467,neutral

Warriors And Soldiers,-5435,sad

Butchers And Soldiers,76542,sad

World Of The Mountain,6536543,sad

Ruination Of Dusk,-2000,happy

Destroying The Stars,5435,happy

Blinded In My Enemies,765745.5,happy
```

Expected output:

```
{'happy': 192421.475, 'sad': 1659412.5, 'neutral': 30733.5'}
```

<u>Example 2:</u>

Text in *in_file* (same as Example 1, but without the *neutral* lines):

```
Descendant Without A Conscience,505.4,happy

Wolf Of The Solstice,30000,sad

Warriors And Soldiers,-5435,sad

Butchers And Soldiers,76542,sad

World Of The Mountain,6536543,sad

Ruination Of Dusk,-2000,happy

Destroying The Stars,5435,happy

Blinded In My Enemies,765745.5,happy
```

Expected output:

```
{'happy': 192421.475, 'sad': 1659412.5, 'neutral': 'NA’}
```

Example 3:

Text in *in_file* (the bold formatting is for explanation purposes, it does not appear in the text):

Descendant Without A Conscience,505.4,happy

Wolf Of The Solstice,30000,sad

Women Of Hope,-4000,neutral

Pirates Of Perfection,65467,neutral

Warriors And Soldiers,-5435,sad

Butchers And Soldiers,76542,sad

World Of The Mountain,6536543,sad

Ruination Of Dusk,-2000,happy

Destroying The Stars,5435,happy

Blinded In My Enemies,765745.5,happy

Women Of Hope,-3000,neutral

Wolf Of The Solstice,30000,sad

In this case the following error message will appear:

ValueError: The series Women Of Hope appears more than once.

and nothing will be returned. Note that the printed series was the one whose second occurrence appears first.

Example 4

Text in *in_file* (the colors are for explanation purposes, they do not appear in the text):

Descendant Without A Conscience,505.4,Happy

Wolf Of The Solstice,30000,glad

Women Of Hope,-4000,

Pirates Of Perfection,a lot money,neutral

Warriors And Soldiers,-5435,sad

Butchers And Soldiers,76542,sad

World Of The Mountain,6536543,sad

```
Ruination Of Dusk,-2000,happy

Destroying The Stars,5435,happy

Blinded In My Enemies,765745.5,happy
```

In this case, the input is not correct, hence the following error message will appear:

```
ValueError: Invalid input.
```