**Task 1:**

The Task 1 takes the form of an "authorship attribution" system, that is a system which attempts to determine who wrote a given document, based on analysis of the language used and style of that document. We will break the system down into multiple parts, to make it clearer what the different moving parts are, and make it easier for you to test your system.

任务1:

任务1采用“作者署名”系统的形式，这是一个系统，它试图根据对文档使用的语言和风格的分析来确定谁编写了给定的文档。我们将系统分解为多个部分，以便更清楚地了解不同的移动部分是什么，并使您更容易地测试您的系统。

**First step:**

The first step in our authorship attribution system will be to take a document, separate it out into its component words, and construct/return a dictionary of word frequencies. As we are focused on the English language, we will assume that "words" are separated by whitespace, in the form of spaces (**' '**), tabs (**'\t'**) and newline characters (**'\n'**).

第一步:

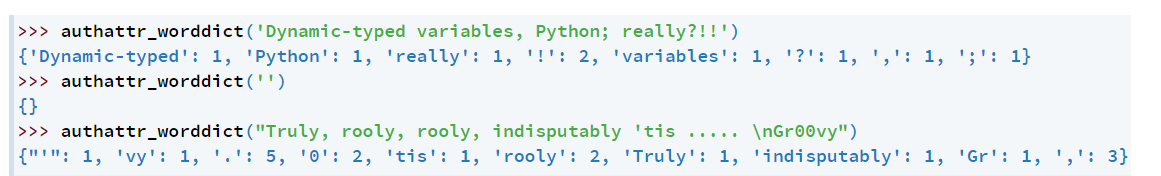
我们的作者归属系统的第一步将是获取一个文档，将它分离成它的组成词，并构造/返回一个单词频率的字典。由于我们关注的是英语，我们将假设“单词”由空格(' ')、制表符('\t')和换行字符('\n')分隔。

We will also do something slightly unconventional in considering each "standalone" non-alphabetic character (i.e. any character other than whitespace, or upper- or lower-case alphabetic characters) to be a single word. For example, given the document **'Dynamic-typed variables, Python; really?!!'**, the component words, in sequence, would be **'Dynamic-typed'**(noting that **'-'** here is not considered to be a word despite being non-alphabetic, as it is surrounded by alphabetic characters), **'variables'**, **','**, **'Python'**, **';'**, **'really'**, **'?**', **'!'**, **'!'**. Note here that, in the case of the document starting with **'Dynamic--typed'**, the breakdown into words would instead be **'Dynamic'**, **'-'**, **'-'**, and **'typed'**, as both of the hyphens neighbor a non-alphabetic letter. Note also that case should be preserved in the output (i.e. if a word is upper case in the original, it should remain in upper case).

我们还将做一些稍微非常规的事情，将每个“独立”的非字母字符(即除空格、大小写字母字符外的任何字符)视为单个单词。例如，给定文档的‘动态类型变量，Python;真的吗? ! !，组成词的顺序将是“Dynamic-typed”(注意，“-”虽然不是按字母顺序排列的，但这里不被认为是一个单词，因为它被字母字符包围)、“variables”、“”、“Python”、“;”、“really”、“?”、“!”、“!”注意，在以“Dynamic- typed”开头的文档中，分解为单词的部分应该是“Dynamic”、“-”、“-”和“typing”，因为这两个连字符相邻的是非字母。还要注意，输出中应该保留大小写(也就是说，如果一个单词在原文中是大写的，那么它应该保留大写)。

Write a function **authattr\_worddict(doc)** that takes a single string argument **doc** and returns a dictionary (**dict**) of words contained in **doc** (as defined above), with the frequency of each word as an **int**. Note that, as the output is a **dict**, the order of those words may not correspond exactly to that indicated below, and that the testing will accept any word ordering within the dictionary.

Here are some example calls to your **authattr\_worddict** function:



编写一个函数authattr\_worddict(doc)，该函数接受一个字符串参数doc并返回doc(如上定义)中包含的单词的字典(dict)，每个单词的频率作为一个int。注意，由于输出是dict，这些单词的顺序可能与下面所示的顺序并不完全对应，并且测试将接受字典中的任何单词顺序。下面是对authattr\_worddict函数的一些示例调用:

**Second step:**

The next step in our authorship attribution system will be to take two dictionaries of word counts and count the similarity between them. We will do this by:

1. ranking the two sets of words in descending order of frequency, and;
2. for corresponding word pairs, calculate the absolute difference in rank between the two.

第二步:在我们的作者归属系统的下一步将采取两个词计数字典，并计算它们之间的相似性。我们将通过:1.将这两组单词按频率降序排列;2.对于对应的单词对，计算它们之间的绝对秩差。

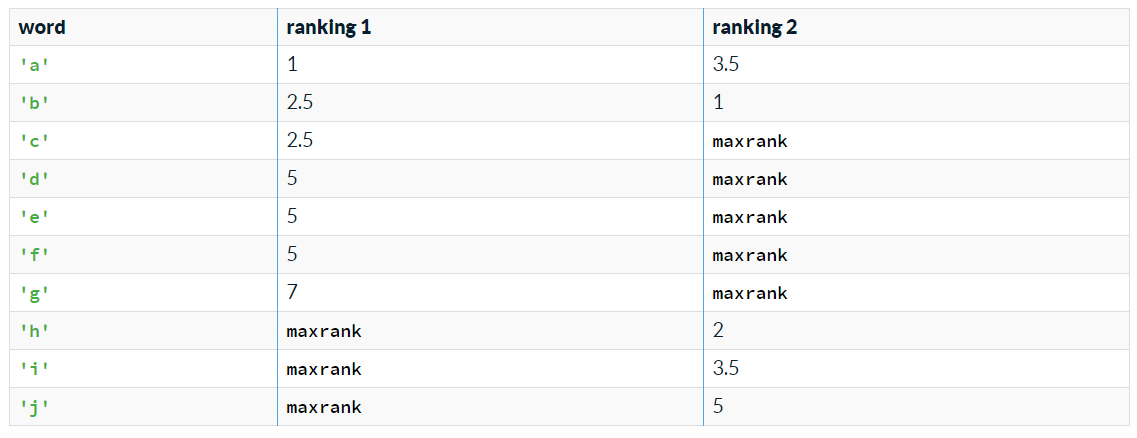
If a word is found in one ranking but not the other, we will set the ranking for the second to the value **maxrank** (provided as part of the function call). In the case of a tie in the word frequency ranking (due to multiple words having the same frequency), we will assign all items the same value, calculated as follows:

如果一个单词在一个排序中找到，而在另一个排序中没有，我们将把第二个排序设置为maxrank值(作为函数调用的一部分提供)。在词频排列为平局的情况下(由于多个单词具有相同的频率)，我们将为所有项目分配相同的值，计算如下:

For example, if two items were tied for second, we would assign each of them the rank . The ranking of the next item would then be 4 rather than 3, as two places in the ranking have been taken. For example, if the first dictionary was **{'a':10, 'b': 5, 'c': 5, 'd': 2, 'e': 2, 'f': 2, 'g': 1}** (i.e. **'a'** occurs 10 times, **'b'** 5 times, etc.), then the corresponding ranking would be:



Note that **'d'**, **'e'** and **'f'** are assigned a ranking of 5 because they are all tied for fourth (three items precede them), and

Then the combined ranking would be:

例如，如果两个项目并列第二，我们将为每个项目分配排名。下一项的排名将是4，而不是3，因为在排名中有两个位置。例如,如果第一个字典是{a: 10 b: 5, ' c ': 5, ' d ': 2,“e”: 2,“f”: 2,“g”: 1}(即。a出现10次，b出现5次等)，则对应的排名为:请注意，“d”、“e”和“f”的排名为5，因为它们都并列第四(前面有3个项目)，而且那么综合排名将是:

**Third step:**

The final step is to calculate the "out-of-place" distance between the two rankings, by calculating the total absolute difference between the respective rankings for each word contained in the union of the rankings ... which is just a complicated way of saying, for each row in the table above calculate the absolute difference between the two ranking values (e.g. for **'a'**, |1−3.5|=2.5), and sum up across all the rows. Assuming that **maxrank** is equal to 10, the value for the case above would be:

|1−3.5|+|2.5−1|+|2.5−10|+|5−10|+|5−10|+|5−10|+|7−10|+|10−2|+|10−3.5|+|10−5|=49.0

第三步:

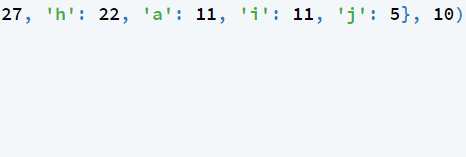
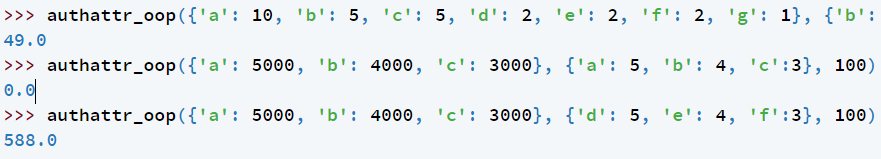
最后一步是计算两个排名之间的“错位”距离，计算出联盟中包含的每个单词在各自排名之间的绝对总差值。这只是一种复杂的说法，对于上表中的每一行，计算两个排序值之间的绝对差值(例如，对于'a'， |1 - 3.5|=2.5)，并对所有行求和。假设maxrank = 10，则上述情况的值为:3.5 | + | 2.5 | 1−−−1 | + | 2.5 10 | + | 5 | + | 5 10−−10 | + | 5 | + | 7 10−−10 | + | 10−2 | + | 10 10−−3.5 | + | 5 | = 49.0

Write a function **authattr\_oop(dictfreq1, dictfreq2, maxrank)** that takes three arguments:

* **dictfreq1**: a dictionary of words, with the (positive integer) frequency of each
* **dictfreq2**: a second dictionary of words, with the (positive integer) frequency of each
* **maxrank**: the positive **int** value to set the ranking to in the case that the word isn't in the dictionary of words in question

and returns a **float** out-of-place distance between the two (where the smaller the number, the more similar the two rankings are).

Here are some example calls to your **authattr\_oop** function:



编写一个函数authattr\_oop(dictfreq1, dictfreq2, maxrank)，它有三个参数:

dicfreq1:单词词典，每个单词的(正整数)频率

dicfreq2:第二个单词字典，每个单词的(正整数)频率

maxrank: positive int value 设置 排名 中 , isn't 这个 词 在 字典 里 的 单词并返回两者之间的浮动错位距离(其中数字越小，两个排名就越相似)。

下面是对authattr\_oop函数的一些示例调用:

The final step in our authorship attribution system will be to perform authorship attribution based on a selection of sample documents from a range of authors, and a document of unknown origin.

You will be given a selection of sample documents from a range of authors (from which we will learn our word frequency dictionaries), and a document of unknown origin. Given these, you need to return a list of authors in ascending order of out-of-place distance between the document of unknown origin and the combined set of documents from each of the authors. You should do this according to the following steps:

1. compute a single dictionary of word frequencies for each author based on the combined set of documents from that author (provided in the form of a list of strings)
2. compute a dictionary of word frequencies for the document of unknown origin
3. compare the document of unknown origin with the combined works of each author, based on the out-of-place distance metric
4. calculate and return a ranking of authors, from most similar (smallest distance) to least similar (greatest distance), resolving any ties in the ranking based on an alphabetic sort

我们的作者署名系统的最后一步将是根据从一系列作者中选择的样本文档和来历不明的文档执行作者署名。

您将从一系列作者(我们将从中学习词频词典)中选择一些示例文档，以及一个来历不明的文档。考虑到这些，您需要返回一个作者列表，按照未知源文档与来自每个作者的文档组合集之间的错位距离的升序返回。你应该按照以下步骤来做:

1. 根据作者提供的一组文档(以字符串列表的形式提供)，为每个作者计算一个单词频率字典
2. 为来历不明的文件计算单词频率字典
3. 根据错位距离度量，将来历不明的文档与每位作者的作品组合进行比较
4. 计算并返回作者的排名，从最相似(最小距离)到最不相似(最大距离)，根据字母排序解决排名中的任何关系

You have been provided with reference implementations of the functions **authattr\_worddict** and **authattr\_oop** from the preceding questions in order to complete this question, and should make use of these in your solution. These are provided via the **from hidden\_lib import authattr\_worddict, authattr\_oop** statement, which must not removed from the header of your code for these functions to work.

Write a function **authattr\_authorpred(authordict, unknown, maxrank)** that takes three arguments:

* **authordict**: a dictionary of authors (each of which is a **str**), associated with a non-empty list of documents (each of which is a **str**)
* **unknown**: a **str** contained the document of unknown origin
* **maxrank**: the positive int value to set **maxrank** to in the call to **authattr\_oop**

and returns a **list** of **(author, oop)** tuples, where **author** is the name of an author from **authordict**, and **oop** is the out-of-place distance between **unknown** and the combined works of **author**, in the form of a **float**.

为了完成这个问题，前面的问题已经为您提供了authattr\_worddict和authattr\_oop函数的参考实现，您应该在解决方案中使用它们。这些函数是通过from hidden\_lib import authattr\_worddict, authattr\_oop语句提供的，要使这些函数正常工作，必须将该语句从代码头中删除。编写一个带有三个参数的函数authattr\_authorpred(authordict, unknown, maxrank):

·authordict: 词典 的 作者 (each a str), 关联 到 一 个 非 空 的 文档 列表 (each a str)

·unknown: a str contained 来历 不明 的 文件

·maxrank: 积极 的 int 值 调用 to authattr\_oop set maxrank to并返回一个(author, oop)元组列表，其中author是来自authordict的作者的名称，oop是以浮点数的形式表示的，是unknown与作者的组合作品之间的错位距离。

For example:

>>> authattr\_authorpred({'tim': ['One One was a racehorse; Two Two was one too', 'How much wood could a woodchuck chuck'], 'einstein': ['Unthinking respect for authority is the greatest enemy of truth.', 'Not everything that can be counted counts, and not everything that counts can be counted.']}, 'She sells sea shells on the seashore', 20)

[('tim', 287.0), ('einstein', 290.0)]

>>> authattr\_authorpred({'Beatles': ['Hey Jude', 'The Fool on the Hill', "A Hard Day's Night", "Yesterday"], 'Rolling Stones': ["(I Can't Get No) Satisfation", 'Ruby Tuesday', 'Paint it Black']}, 'Eleanor Rigby', 15)

[('Beatles', 129.0), ('Rolling Stones', 129.0)]

例如:

('tim': 'One One是一匹赛马;“Two Two was one too”，“一只土拨鼠能扔多少木头”，“einstein”:“不加思索地尊重权威是真理最大的敌人。”’、‘不是每件可以计算的东西都有价值，也不是每件可以计算的东西都有价值。，“她在海边卖贝壳”，20)[(“tim”，287.0)，(“einstein”，290.0)]>>> authattr\_authorpred('披头士':['Hey Jude'， 'The Fool on The Hill'， ' A Hard Day's Night '， "Yesterday")， 'Rolling Stones': ["(I Can't Get No) Satisfation"， 'Ruby Tuesday'， 'Paint it Black']]， 'Eleanor Rigby'， 15)[(披头士乐队，129.0)，滚石乐队，129.0]

**Task 2:**

读文件tox21.csv , 该文件为毒性分子数据库，其中O列smiles属性表示该分子式，B-M行表示该分子具有的12中化学属性，1表示有该属性，0表示无该属性，没有数字表示未知。

根据tox21.csv生成一个新的.csv文件，该新文件包含12个sheet，每一个子文件表示一种属性。第一列为分子序号（与tox21.csv中的第A列序号一致），第二列为Mol\_id(与tox21.csv中的N列内容一致)，第三列为分子式，第四列为是否具有该属性，若属性值未知，则该分子不包含在当前sheet中。

**Task 3:**

SQL在线学习，<https://sqlbolt.com/>，完成所有训练。