**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采用“作者归属”系统的形式，即根据对所用语言和文档样式的分析，试图确定谁编写了给定文档的系统。我们将把系统分解为多个部分，以便更清楚地了解不同的运动部件是什么，并使您更容易测试系统。

**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'**).

我们的作者归属系统的第一步将是获取一个文档，将其分离为组成词，并构造/返回一个词频率字典。当我们关注英语时，我们假设“单词”由空格（“”）、制表符（“\t”）和换行符（“\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).

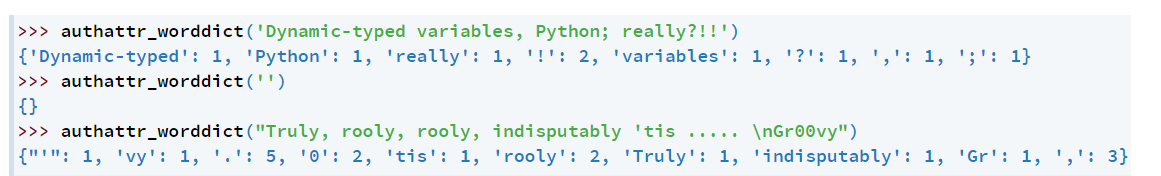
在考虑每个“独立”的非字母字符（即除空格以外的任何字符，或大写或小写字母字符）是单个单词时，我们也会做一些稍微不合常规的事情。例如，给定文档的动态类型变量python；真的吗？！！，按顺序，组成词将是“动态类型”（注意，这里的“-”虽然不是字母，但不被视为一个词，因为它被字母字符包围）、“变量”、“、”、“python”、“；”、“真的”、“？”，“！”，“！”。请注意，对于以“dynamic--typed”开头的文档，分解为单词将改为“dynamic”、“-”、“-”和“typed”，因为两个连字符都是非字母。还请注意，大小写应保留在输出中（即，如果单词在原始单词中为大写，则应保留在大写）。

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），它接受一个字符串参数Doc，并返回一个包含在Doc（如上定义）中的单词字典（Dict），每个单词的频率作为一个int。注意，由于输出是一个dict，这些单词的顺序可能与下面所示的顺序不完全对应，并且测试将接受字典中的Y字排序

以下是对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;

for corresponding word pairs, calculate the absolute difference in rank between the two.

作者归属系统的下一步将是采用两个词数词典，并计算它们之间的相似性。我们将通过以下方式做到这一点：

1.按照词频降序排列

2.对于对应的词对，计算两个词之间的绝对秩差。

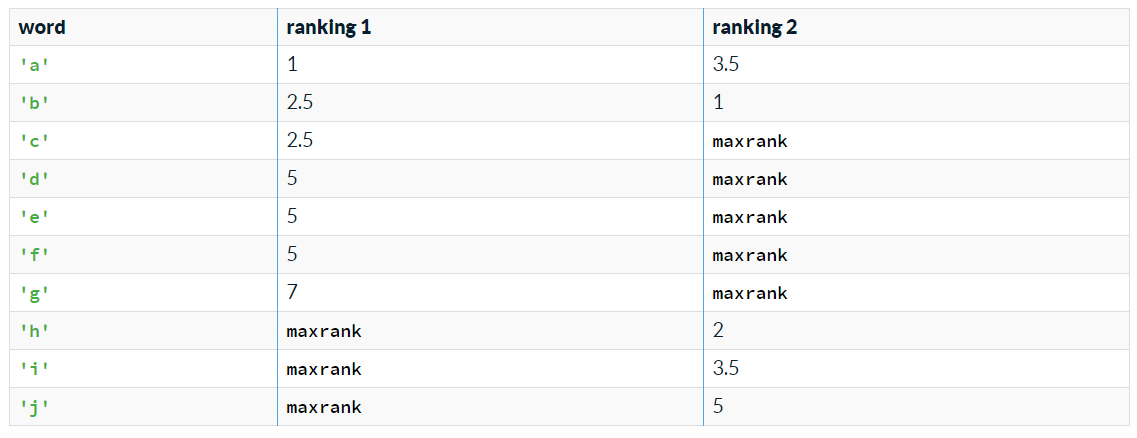
If a word is found in one ranking but not the other, we will set the ranking for the second to the value **maxrank** (provided as part of the function call). In the case of a tie in the word frequency ranking (due to multiple words having the same frequency), we will assign all items the same value,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

如果在一个排名中找到一个单词，而不是另一个，我们将把第二个单词的排名设置为值maxrank（作为函数调用的一部分提供）。如果词频排序中出现并列（由于多个词具有相同的词频），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:

**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:

最后一步是计算两个排名之间的“错位”距离，通过计算各个排名之间的绝对差，将每个单词包含在排名的并集中…这只是一个复杂的说法，对于上表中的每一行，计算两个排名值之间的绝对差（例如，对于“a”，1−3.5=2.5），并在所有行中求和。假设maxrank等于10，则上述情况的值为：

|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

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

•dictfreq1：单词字典，每个单词的（正整数）频率

•dictfreq2：第二个单词字典，每个单词的（正整数）频率

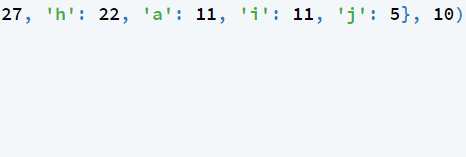
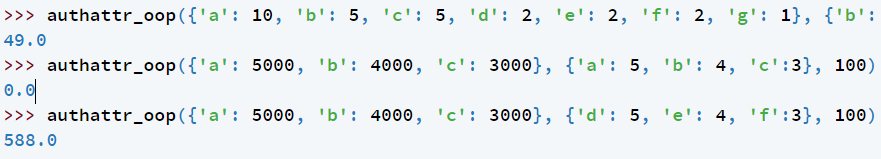
•maxrank：当单词不在相关单词字典中时，将排名设置为的正in t值。

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:

并返回两个排名之间的浮动位置距离（数字越小，两个排名越相似）。

以下是对AuthAttr ou 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:

作者归属系统的最后一步是根据从一系列作者中选择的样本文档和来源未知的文档执行作者归属。

我们将为您提供一系列作者的示例文档（我们将从中学习词频词典）和一份来源未知的文档。考虑到这些，您需要按升序返回一个作者列表，在未知来源的文档和来自每个作者的组合文档集之间的不适当距离。您应该按照以下步骤执行此操作：

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

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.计算未知来源文件的词频字

3.根据异地距离度量，将来历不明的文献与各作者的合著作品进行比较。

4.计算并返回从最相似（最小距离）到最不相似（最大距离）的作者排名，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，您已经从前面的问题中获得了AuthAttr\_WordDict和AuthAttr\_op函数的参考实现，并且应该在解决方案中使用这些函数。这些是通过from hidden\_lib import authattr\_worddict、authattr\_oop语句提供的，这些语句不能从代码头中删除，以使这些函数正常工作。

编写一个函数authattr\_authorpred（authordict，unknown，maxrank），它接受三个参数：

•authordict：与非空文档列表（每个文档都是str）关联的作者词典（每个文档都是str）

•未知：str包含未知来源的文档

•maxrank：调用authattr ou oop时要将maxrank设置为的正int值。

并返回（author，oop）元组的列表，其中author是authordict中的作者的名称，oop是未知和作者组合作品之间的不适当距离，形式为浮点。

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)]

**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。