Instructions

SPECIFICATIONS:

1. Data sources

a) Read two parameters from the command line with the names of two files you have obtained from Project Gutenberg. If the command line does not have exactly two parameters, display an error message and quit.

Test files are:

Pride and Prejudice in UTF-8:
https://www.gutenberg.org/files/1342/1342-0.txt

Du côté de chez Swann in UTF-8:
https://www.gutenberg.org/files/2650/2650-0.txt

You can learn more about Unicode in Python here:
https://docs.python.org/3/howto/unicode.html

b) Whenever you read a file, you need to print appropriate information at the end of your program so that you can track back and verify *from an external source* that your input is correct.

For this program, appropriate data to print for each file would be the number of lines, number of characters and number of letters (alphabetic characters) in the file, along with appropriate rubrics. Use appropriate functions from the string class.

c) An appropriate source for the number of lines and number of characters would be the Linux operating system. There are several ways to determine the number of lines and number of characters in a file. How many can you figure out?

d) The number of letters is a good check for the rest of your code. A check between two parts of a program is not as strong as validation from an external source but is still a worthwhile thing. In a real-world context, there are Unix utilities that you could use to validate this number directly, but it would probably be both easier and more worthwhile to validate selected numbers from the detail section of the program, e.g., the number of ‘a’s and ‘â’s in the file.

In a high-stakes real-world context, it’s also possible to assign two separate programming teams that are required to work independently (the “black-box” approach). It is rumored that this was done during the American space program in the 1960’s.

2. Calculate the following information while reading each file one record at a time:

a) Convert the text to lower case, again using appropriate functions of the string class. This is a standard procedure in natural language processing to obtain a canonical form of words.

b) Count the number of instances of vowel contained in the file. We will consider the vowels to be a e i o u y, the French vowels with diacritics listed below, and the two compound vowels listed below. Use a dictionary to store the totals.

U+00E9 = é (e with acute accent)
U+00E2 = â (a with circumflex)
U+00EA = ê
U+00EE = î
U+00F4 = ô
U+00FB = û
U+00E0 = à (a with grave accent)
U+00E8 = è
U+00F9 = ù
U+00EB = ë (e with dieresis)
U+00EF = ï
U+00FF = ÿ
U+00FC = ü
U+00E6 = æ (ligature or digraph)
U+0153 = œ

Most of these characters have capital-letter equivalents, but these are not relevant to the assignment due to the processing specified above.

A consonant is any letter that is not a vowel. French has one consonant with a diacritic, the “c cedilla”, ç (U+00e7)/Ç (U+00c7), but that information isn’t needed for this assignment.

3. Use the chi-square statistic to investigate the hypothesis that one of these texts has a greater percent of vowels than the other. Look at the number of consonants and the number of vowels for each file. Print the contingency table, the value of the chi-square statistic and the number of degrees of freedom. Write a general-purpose chi-square function that you will be able to use in the next step also. In general, it’s a good idea to do calculations in functions but print from the process that calls the function (which could be another function, of course). That allows the caller to decide whether and how to print the output.

Store the data in a 2-D array implemented as a list of lists and use list comprehensions to process it.

When printing, data should always have appropriate rubrics. Integers should always be right-aligned and floating point numbers should always be decimal-aligned. For this assignment, all percents should be rounded to 2 decimal places.

4. A second experiment

a) Build a dictionary that can be used to convert from a letter with a diacritic, e.g., ‘â’, to its equivalent without diacritics, which in this case would be ‘a’.

b) As you read each line, also make a version without diacritics, e.g., converting ‘é’ to ‘e’. Use the dictionary you built above. For this purpose, also convert ‘æ’ to ‘ae’ and ‘œ’ to ‘oe’. That will change your total letter count. Use a dictionary to keep track of the quantity of each vowel in the deaccented version.

c) Use the chi-square statistic to investigate the hypothesis that the distribution of vowels is different in these two texts. For this purpose, you will have to use the deaccented forms to make the comparison meaningful, i.e., you are comparing the frequencies of a, e, i, o, u and y in the English text versus their frequencies in the deaccented form of the French text. Print the contingency table, the number of degrees of freedom, and the value of the chi-square statistic for this experiment. Call the same function that you wrote for the first experiment.

5. As is standard in computer science, you may only read the file *once*. In addition, do not repeat code unnecessarily: make functions where appropriate. Some of these functions will return multiple values.

6. Make sure you know the answers to the following questions:

a) Which Unix utilities can you use to validate the number of lines or number of characters in a file?

b) Were the results of the first experiment significant at the 5% level? At the 1% level? What does “significant at the 5% level” mean?

c) Same question for the second experiment.

Output Sample:

(this is a contingency table, although these are not the real numbers)

```
Actual:

Book            Consonants          Vowels
abc.txt            341985           209729
def.txt            438211           363792

Expected:
Book            Consonants          Vowels
abc.txt         317972.70           233741.30
xyz.txt         462223.30           339779.70
```

df = 1
chi-square = 7224.50

(contingency table for part 2)

Actual:

Book	a	e	i	o	u	y
abc.txt	etc.					
def.txt						