

Assignment 4

Goals

The goal of this assignment is to work with files, iterators, strings, and string formatting in Python.

Instructions

You will be doing your work in a Jupyter notebook for this assignment. You may choose to work on this assignment on a hosted environment (e.g. [tiger](#)) or on your own local installation of Jupyter and Python. You should use Python 3.9 or higher for your work. To use tiger, use the credentials you received. If you work remotely, make sure to download the .ipynb file to turn in. If you choose to work locally, [Anaconda](#) is the easiest way to install and manage Python. If you work locally, you may launch Jupyter Lab either from the Navigator application or via the command-line as `jupyter-lab`.

In this assignment, we will be working with texts from [Project Gutenberg](#) which is a library of eBooks which includes a lot of literature for which U.S. copyright has expired. In this assignment, you will process these files, count vowels and consonants, and convert some of the text. We will use not only English texts but those from other languages to demonstrate the benefits of Unicode encoding.

A [template notebook](#) is provided with a cell to help download the files given a URL. You will read this data, do some calculations, and write a new output file in a similar format. We will be working with three books, but write your code so that it can be adapted to work with other texts as well. The three texts are:

- *Pride and Prejudice*, J. Austen. <https://www.gutenberg.org/files/1342/1342-0.txt>
- *Du côté de chez Swann*, M. Proust. <https://www.gutenberg.org/files/2650/2650-0.txt>
- *Sense and Sensibility*, J. Austen. <https://www.gutenberg.org/files/161/161-0.txt>

Due Date

The assignment is due at 11:59pm on Wednesday, March 9.

Submission

You should submit the completed notebook file required for this assignment on [Blackboard](#). The filename of the notebook should be `a4.ipynb`. You **should not** turn in the output files (Part 5) as your notebook should contain the code to create it.

Details

Please make sure to follow instructions to receive full credit. Use a markdown cell to **Label** each part of the assignment with the number of the section you are completing. You may put the code for each part into one or more cells. Do not use external libraries for reading, parsing, or writing the data files.

0. Name & Z-ID (5 pts)

The first cell of your notebook should be a markdown cell with a line for your name and a line for your Z-ID. If you wish to add other information (the assignment name, a description of the assignment), you may do so after these two lines.

1. Read Data (10 pts)

Use the `download_text` function in the provided [template notebook](#) to download files. Then, write a function that given the filename, opens the local file and reads the downloaded file, returning the processed lines as a list. Project Gutenberg includes metadata at the beginning of the text and license information at the end of the text. The actual text is contained between the lines

```
*** START OF THE PROJECT GUTENBERG EBOOK <TITLE> ***
```

and

```
*** END OF THE PROJECT GUTENBERG EBOOK <TITLE> ***
```

where `<TITLE>` is the name of the book. Read (and discard) lines until you hit the *START* tag. Then, read in all the lines up to but not including the *END* tag into a list of strings. Remove any leading and trailing whitespace from the lines. You will need to figure out how to find the *START* and *END* tags and which lines should be included.

Hints

- You may check for the tags using a regular expression, but python's string methods should also work.
- If you pass an iterator to a for loop, it will loop through the **remaining** items.
- `strip` will be useful to remove whitespace.

2. Count Vowels and Consonants (15 pts)

Next, write a function to loop through the list of lines to count the vowels and consonants in the texts. For our purposes, we will count any of the letters in the string “aeiouyàâæèéêëïîôùûüÿœ” (**updated 2022-03-02** to add “y”) as vowels. A consonant is any **alphabetic** character that is not a vowel. You will need to convert the lines to lowercase before checking whether the individual characters are vowels or consonants or something else, like punctuation. Return both the number of vowels *and* the number of consonants

Hints

- Remember that you will need to iterate through each **character**
- Check the `is*` functions in the string class to check for letters.

3. Convert Diacritics and Digraphs (15 pts)

Next, write a function that creates a new version of the text where the vowels with diacritics (àâèéêëïîôùûüÿ) are converted to versions without them (e.g. á becomes a, î becomes i). In addition, the digraphs æ and œ should be expanded to ae and oe, respectively. (**Updated 2022-03-02**) Also, make sure to include the uppercase variants of all these conversions, but use a programmatic way to do this from the lowercase conversions. Your output should be a new list with the converted lines of text. You **may not** use a series of if or elif statements for this task.

- Consider using a dictionary to map the conversions
- Remember for lower/uppercase variation that you can merge two dictionaries
- Comprehensions can be useful in applying the modifications to each character
- You can join using an empty string

4. Write Output (20 pts)

Write the converted text (from Part 3) to a file. In addition, put your name at the beginning of the file as the converter, then a line marking the start of the text (***** START OF THE EBOOK *****), the converted text, and then a line marking the end of the text (***** END OF THE EBOOK *****). Finally, use your function from Part 2 to compute the number of vowels and consonants before **and** after conversion. Compute the change as the value after minus the value before divided by the value before. Multiply by 100 to obtain a percentage and print with a + or - sign depending on the change. (Use the format mini-language for this.) The statistics need to be right-aligned so your output should look like (**updated 2022-03-02**):

```
From Project Gutenberg (www.gutenberg.org)
Converted by: Iamma Huskie

*** START OF THE EBOOK ***

...

*** END OF THE EBOOK ***

Number of vowels before:      360122
Number of vowels after:      360385
Conversion Change:           +0.07%

Number of consonants before:  426580
Number of consonants after:   426580
Conversion Change:           +0.00%
```

The name of the file should be the original filename with `-converted` added. For example, `2650-0-converted.txt`.

Hints

- Use `print` function calls to write a few data items to `stdout` first, then when you are satisfied the format is correct, write all of the data items to a file.
- Remember the file keyword argument for the `print` function.
- Use a `with` statement to make sure all data is written to the file (or make sure to call `close`).
- Remember to pass the `w` flag to `open` to be able to write to a file.
- Consult the [Format Specification Mini-Language](#) for the various flags
- You can see the contents of the file you wrote in the notebook using the `!cat 2650-0-converted.txt`

5. [CSCI 680 Only] Compare Texts (15 pts)

Now, we wish to test the differences between two texts, specifically to investigate the hypothesis that one text has a greater percent of vowels than the other. For this part, write a single function that takes two urls and uses the functions written in Parts 1-3 to compute the contingency table showing the number of vowels and consonants for each text with their totals. Make sure to do the conversion in Part 3 as well. Right-align the results. Then, compute the [chi-squared test](#) for the vowels and consonants and test it on *Pride and Prejudice*

vs. *Du côté de chez Swann* (two different languages) and *Pride and Prejudice* vs. *Sense and Sensibility* (same language, same author). Report the p-value. Use the function [scipy.stats.chi2_contingency](#) to do this calculation. Note that it does not need the totals, only the upper-left 2x2 square. Sample results (**updated 2022-03-02**):

```
>>> compare_texts('2650-0.txt', '1342-0.txt')

2650-0.txt      360385      426580      786965
1342-0.txt      216806      320179      536985
    Total      577191      746759     1323950

p-value: 0.0

>>> compare_texts('161-0.txt', '1342-0.txt')

          # of Vowels  # of Consonants      Total
161-0.txt      212215      313598      525813
1342-0.txt      216806      320179      536985
    Total      429021      633777     1062798

p-value: 0.8740327835853077
```

Hints

- Formatted string literals will be useful for this part of the assignment.
- Consult the [Format Specification Mini-Language](#) for the various flags

Extra Credit

- CSCI 490 Students may complete Part 5 for extra credit
- All students may add the book name into the *** lines in Part 4 (5 pts)
- All students may add the book name to the contingency tables in Part 5 (5 pts)