Assignment: News Editor, Final Project in Python

After completing this assignment, a student can:

- Create a Python program to perform functions to support/fulfill program requirements
- Print text to illustrate the result
- Run code to verify that it works as intended (WAI)

Lab Assignment Prompt: News Editor-in-Chief

As the news editor, who also happens to have some Python skills, your role requires you to review your reporters' work each week to make sure it communicates clearly to your audience. Some of this work is repetitive - especially finding those words and phrases that authors tend to frequently reuse.

To save yourself some time and complete the weekly review faster, you will create a python program to:

- Inspect a text document provided by any author/reporter.
- Count the number of times each word appears in the text. (Common articles, such as a, an, the, may optionally be ignored)
- Search for a specific word and replace it with something different.
- Open a file, operate upon, and save a new version of, a text file stored locally on your computer.

NOTE: Run **ALL** of the code you created and make sure it **works as intended** (WAI).

Program Requirements:

Import a document

- Count the number of total words in the document.
- Count the number of each word in the document and store the word and its count in a dictionary, with the word as the Key and the count as the Value.
  - NOTE: this requirement is basically the dictionary counting trick.
- Include a way to exclude common but unimportant words, such as "a", "an", "the". The user of your code can input a list of these unimportant words and there can be as many as the user would like to ignore.
- Store a second dictionary with each word's proportion as a float with four decimals, of the text's total words. For example, if the word, "weather" is found 143 times out of 5390 total words, the proportion will be calculated as 143 / 5390, and the value will be 0.0265.
  - NOTE: this requirement is basically a twist on the dictionary counting trick, so students can't just copy the counting trick. But they can get close to this.
- Accept a user's query of whether a word exists in the text. If the word search finds a word, state the word, the number of times it appears, and the proportion of the total words made up by that word. If the word does not find the word, return some information that the word was not found in the text.

Assignment continues on the following page.

- Accept a user's request to list the top **n words** in the text. (The user chooses how many.) The code will print the word, the absolute count, and the proportion of those words - each on its own line.

Modify the document

- Search for a specific string of text within the document, and from this search:
    - Store as a list the position(s) of the string within the text. For example, if the string appears as the 50th word, the dictionary will include the value 50 as one of the list items. The list should be sorted from least to greatest (ascending order).
    - Allow the user to specify a replacement text string. For example, if the user searches for "weather", but would like to replace the term with "whether".
    - Allow the user to choose either one or all examples of the positions to replace. For example, if "weather" appears as the 50th, 132nd, and 533rd words in the text, the user can replace the term found in position 50. Or, if the user would like to replace all instances, this can be done instead.

Save the output file as a new file

- Accept the name of the original file.
- Accept a name for the new file.
- Require that the new name be different from the original name.

Reusability

- Any user should be able to reuse your code to do the same thing with any text
- All code must be 100% encoded within functions, modules, or classes
- Functions must perform discrete and intuitive tasks required by this assignment - don't make a single function that does it all.

REMINDER: Run all of the code you created and make sure it **works as intended** (WAI) and **Review Assignment Rubric (on pages 4-6).**

Assignment continues on the following page.

Assignment Upload (Submission) to Blackboard:

1. From the command line at the root directory of your project, execute the following command:

   ```
   git bundle create firstname-lastname-NEWS-date.bundle –all
   ```

   Note: firstname and lastname are your first and last names, and date is in the format of mmddyyyy.

2. Access your Blackboard course (https://courses.cscc.edu/)
3. Open the course and navigate to the assignment.
4. Click **UPLOAD** in the title of the assignment
5. Click Browse Local Files and attach your file
6. Upload (**SUBMIT**) the file to the corresponding assignment link in Blackboard.

Final Project Scoring Guide				
How am I doing?	**Needs Improvement 0-19 pts**	**Emerging 20-31 pts**	**Proficient (80%+) 32-40 pts**	**Score /40**
Code Functionality **Key Requirements:** • COUNTS AS THREE REQUIREMENTS: All code is 100% encoded within functions, modules or classes. • Functions perform the following tasks: • Accept the name of a document and import the document. • Count Total words in the document and include a way to exclude common but unimportant words.	• Fewer than 7 requirements delivered as expected. • 30 Points deducted if lab produces a syntax error that prevents the lab from running.	• 7 - 10 requirements delivered as expected. • 15 points deducted if lab produces a runtime error but still runs under certain circumstances.	• 11 - 13 requirements delivered as expected.	

- Count the number of each word in the document and store the word and its count in a dictionary, as well as that word's proportion to the total document word count, as a float with 4 decimals.
- Accept a user's query of whether a word exists in the text.
- If the user's query of a word finds the word, Print the word, Print the number of times it appears, and Print the proportion of the total words made up by that word.
- Print the top n words (n chosen by the user). Print the words, Print the number of times each word appears, and Print the proportion of the total words made up by each word.
- Search for a specific string of text within the document, and list the position(s) of the string within the text, sorted from least position to greatest position in the text (ascending order).
- Provide a way to replace a text string that exists in the document with another text string provided by the user. Be able to replace Any or All instances of the original text.
- Save the modified file as a new file. Accept a new name for the new file, which must be different from the original file name.

Final Project Scoring Guide				
How am I doing?	**Needs Improvement 0-19 pts**	**Emerging 20-31 pts**	**Proficient (80%+) 32-40 pts**	**Score /40**
Code Structure	• Script is not simple and is overly complex, AND • Several or many unnecessary redundant operations, AND/OR • No functions not use the return keyword but should; AND/OR • Functions do not use arguments as inputs but should.	• Script is not simple or is overly complex, OR • Some unnecessary redundant operations, OR • Some functions do not use the return keyword but should; AND/OR • Some functions do not use arguments as inputs, but should.	• Script is simple and not overly complex • Few or no unnecessary redundant operations • All functions use the **return** keyword EXCEPT the function that saves a new file; or except obvious examples where return is not practical. • The functions use arguments as input where appropriate.	

How am I doing?	**Needs Improvement 0-9 pts**	**Emerging 10-15 pts**	**Proficient (80%+) 16-20 pts**	**Score /20**
Code Quality & Readability	• Code is difficult for humans to read or understand AND • PEP-8 validation produces multiple hard warnings or errors AND deviations from PEP-8 do not improve code clarity AND/OR	• Some code is difficult for humans to read and understand, OR • PEP-8 validation produces one or two hard warnings or errors AND/OR deviations from PEP-8 do not	• All code is easy for humans to read and understand. • PEP-8 validation produces no hard warnings or errors and Deviations from PEP-8 improve code clarity.	

• Some unnecessary code artifacts are left in the assignment, AND • Some comments are excessive or unnecessary, AND/OR • Variable naming styles are inconsistent, AND/OR • Docstrings are missing.	improve code clarity. AND/OR • Variable naming styles are inconsistent, AND/OR • Docstrings are missing.	• No unnecessary code artifacts are left in the assignment. • All comments are appropriate and provided valuable clarity. • Variable naming styles are consistent. Docstrings are present.