Library of Congress -- CS 1420 Version

Read all Instructions carefully. Not following instructions will result in you not getting the credit you want for this assignment.

Learning Outcomes

1. Read structured input from text files
2. Write structured output to text files
3. Process structured strings
4. Use data structures such as list, tuple, dict in programs
5. Sort data
6. Rearrange data to make processing easier
7. Compute basic statistics or metadata from data
8. Test your code, especially functions, with `doctest`

Problem to Solve

You're an intern at the Library of Congress. You are given a collection of text files where each file corresponds to a book. Each file is named with a three-letter code for the book it contains. Examples are like `TTC.txt`, `WOO.txt`, `TTL.txt` and other names too. There could be many such input files and we may not know how many.

Each line in the file looks like this: one line from the book and the line number, separated by a '|'. For example: "It was the best of times, it was the worst of times,"|1 You discover though, that the content of each file is scrambled so that the lines are not in order.

Your task is to write a program that reads each of these book files and does a few things:

1. Find the longest line from the book along with its line number.
2. Calculate the average length of the lines in the book.
3. Unscramble the lines of text in order of line number.

Once your program has done that, it needs to save this information in a new file called '<three_letter_code>_book.txt'. The format would be:

1. first line is the three-letter code
2. second line is the longest line with line number and content as shown
3. third line is the average length
4. lines 4 to end are the contents of the book in line order

As an example, for this command line input

```
python3 library.py TTL.txt
```

Your code will generate a file named as `TTL_book.txt` and the first few lines of that output file would look like this:

```
TTL
Longest line (14153): Ecclesiastes—"Who can fathom the soundless depths?"—two men out of all
Average length: 58
Twenty Thousand Leagues Under the Seas
An Underwater Tour of the World
JULES VERNE
[......]
```

If there are multiple lines with the same length as the longest, use the line number to decide: the line that appears **later** is considered as "longest". Round the average to the closest integer.

And that's it! You've gathered useful information, unscrambled the text, and written the information to a new file.

Implement the program in a file called `library.py`.

How To Test

- Test with the different input files you are given.
- Test with a sample file that is small enough that you can verify the correct results by hand.
- If your results do not match expected results, most likely you have misunderstood some requirement, made an incorrect assumption, or implemented some computation incorrectly. It is unlikley that Python is broken. Get help.
- Write and run doctests for each function where it makes sense to do it.

If you design functions in your code using the **one function one job** rule, it should be straightforward to see how to use `doctest` to test those functions. For example, you could write one grand function that does the longest line, shortest line and average length and anything else you need. That's harder to test than one function that does each of those things separately and which you can test independently.

Correctness

Each time you submit your assignment, GitHub will run some basic checks on your code. You may submit as many times as needed up to the due date.

Style

In your terminal, execute the below to evaluate the style of your code using `ruff`.

`ruff format library.py` `ruff check library.py`

How to Submit

From your Codespaces workspace, when ready to submit, click the Commit button and write a meaningful Commit message.

Grading

Criterion	100 (Mastery)	85 (Proficient)	70 (Basic)	50 (Needs Improvement)
Produces an output file for each book file given	Program consistently produces an output file for each book file given	Program produces an output file for most book files given	Program produces an output file for some book files given	Program does not produce an output file for book files given
Input through one command line parameter	Input is consistently given through one command line parameter	Input is mostly given through one command line parameter	Input is sometimes given through one command line parameter	Input is not given through one command line parameter
Produces correct text for another book file	Program consistently produces the correct text for another book file not given	Program mostly produces the correct text for another book file not given	Program sometimes produces the correct text for another book file not given	Program does not produce the correct text for another book file not given
Main function with conditional execution	Program contains a main function with clear conditional execution	Program contains a main function with some conditional execution	Program contains a main function but lacks clear conditional execution	Program does not contain a main function with conditional execution
No global code	There is no global code; all code is within functions	Minimal global code; most code is within functions	Some global code; some code is within functions	Significant global code; most code is not within functions
One function one job rule	Each function performs a single, clear task	Most functions perform a single, clear task	Some functions perform a single, clear task	Functions perform multiple tasks
Using doctest snippets where it makes sense	Doctest snippets are used effectively where appropriate	Doctest snippets are used but not always effectively	Doctest snippets are used but inconsistently	Doctest snippets are not used
Passes `ruff check` for syntax errors, violations, and potential bugs	Code passes all `ruff check` for syntax errors, violations, and potential bugs	Code passes most `ruff check` for syntax errors, violations, and potential bugs	Code passes some `ruff check` for syntax errors, violations, and potential bugs	Code does not pass `ruff check` for syntax errors, violations, and potential bugs

Students should strive for mastery level. Lower than that indicates areas where more practice is needed or more learning is needed. Code score is the average of the individual feature scores.

Total score is 1/4 * style score + 3/4 * code_score.