CBS3947 Programming assignment 2 (Due Feb 23 Wed 10:30am)
Total = 14 pts
This is a group assignment. Each group only need to submit ONE assignment. There can be at most THREE students in each group. You may choose to complete this assignment alone, but the grading criteria will be the same for individual and group work.

Task: Recognize positive and negative reviews

In this task, you will design a program that can distinguish positive and negative product reviews. Your code will open a .txt file with multiple paragraphs of reviews (each paragraph representing a different review), and your code will decide whether each review is overall a positive review (i.e., saying good things about the product) or a negative review (i.e., saying bad things about the product). You are recommended to use Spyder both for coding and for viewing plain text files. Save your code with the filename `PA2.py`. This is the only file you need to submit for this assignment.

To help you with the task, you can find in the attachment a sample review text file (`Reviews.txt`), and two word lists (`positive-words.txt` and `negative-words.txt`). The sample review file contains 10 reviews about a camera product, taken from a shopping website and written by different buyers. The two word lists, as the file names suggest, contain a list of positive words and a list of negative words, respectively.

Before you start working on the code, please take some time to go through the sample review file and the word lists. As a human reader, which reviews do you think are positive and which are negative? How do you make the decision? What linguistic features do you think are important in giving people the idea about the polarity (i.e., positive or negative) of the review? When you go through the word lists, do you agree that these words are generally accepted as positive/negative words? (You may notice that the word lists also contain words that are misspelled – this is because misspelling is very common on the Internet and we want to capture all possible spellings of positive and negative words.)

Now, assuming that you have read through the .txt files, you can start thinking about how your code will decide if a review is positive or negative. Obviously, a very simple idea is to just count how many positive words and negative words appear in a review. If more positive words are used than negative words, the review is positive; otherwise, the review is considered negative. Would this simple idea work? In this programming assignment, your basic task is to implement a code that uses this simple algorithm (i.e., counting the number of positive and negative words in a review).

Below is an example of (incomplete) pseudocode, just to give you some ideas of the general structure of your code.

```
# This program reads in a review text file, and outputs the
polarity of each review.

# Open the review file

# Read from the positive and negative word files, and save the
words in two lists

# For each review, count and compare the numbers of positive
words and negative words

# If there are more positive words than negative words, the
review is considered positive. Otherwise, the review is
considered negative.


# Print out the results
```

1. Basic requirements

As mentioned above, your code should implement the simple algorithm of counting and comparing the numbers of positive and negative words in a review and making a judgment based on the result of the comparison. This is the most important requirement. Although the general algorithm is already given, this programming task does require attention to a lot of details. Is your code reading the content of the .txt files properly? Do you need to be concerned about line breaks when reading from the files? Do you need to be concerned about case when processing the texts? Do you need to be concerned about punctuations when looking for positive or negative words? The success of your code relies on the proper handling of these issues. You are recommended to follow incremental programming and test your code bit by bit.

Secondly, your code should also have a reasonable user interface. When the user runs your code, the user will first be asked to input the name of the review file. This also means that we can test your code with a similarly-formatted text file that is **not** `Reviews.txt`. When the program finishes, the user should see informative output in a reader-friendly format. For example, just printing "positive, negative, negative, positive, …" to the screen is not friendly to the user, as the user doesn't know which reviews are associated with which judgment. Also, users may prefer to see not only the judgment but also some explanations as to why a certain review is considered positive or negative. Take this into consideration when designing your program.

Lastly, your code should contain sufficient comments to explain your ideas. Importantly, when you see the output results from your code working on `Reviews.txt`, do the results match

with your own ideas about the positive/negative polarity of each review? Does your code do a good job? Write down your thoughts in a comment section at the end of your code.

For the basic requirements of this task, your code should only use the built-in Python functions and not any imported packages.

2. [Optional] Advanced requirements

This part is optional. Some of you might feel unsatisfied with such a simple algorithm. Can you point out some problems this simple algorithm may have, and/or propose some ways to make this algorithm better? Apart from the number of positive/negative words, anything else that should/could be considered? Can you try to implement some ideas too?

If you are attempting to do the advanced requirements, write your thoughts in a separate comment section titled "advanced requirements" at the end of the code. In this section, you can also tell us where in the code you have already implemented the additional mechanisms.

3. Grading rubrics

This assignment will have a total of **14 points**. Grading criteria are as follows:

Criteria	**Points**
The program is readable. There are sufficient comments in the program for human readers to understand the program design.	4'
The program runs without errors. When we test your program in Spyder, there shouldn't be any error message. If you cannot get the program to do everything you want it to do, it is better to have a program that does less but works than a program that attempts to do more but breaks. You can use comments to include and explain the code you wanted to include but generate errors that you couldn't successfully debug.	5'
The program works reasonably well for the basic requirements	5'

You will get maximally 3 extra points for the advanced requirements, depending on how much you do.

4. Submission

Please submit your program `PA2.py` to this Blackboard assignment by the deadline**.** Since this is a group assignment, each group only needs to submit once. **The person who submits for the group should indicate in the submission the names of the other group members.**