# Project 1
# Nerf herder!
**Due November 18, 2022 at 5pm**
**(5% bonus for submission before 5pm Nov 17)**

Project 1 is the first assignment where we won't be telling you exactly what to do! You will get the chance to make your own decisions about how you want your program to handle things, what its functions should be called, and how you want to go about designing it. Project 1 will also be ***substantially longer*** than any of the single homework assignments you've completed so far, so make sure to take the time to plan ahead.

In order to encourage you to start planning (and implementing) during the first week of the assignment, you have a **mandatory design review due before 5pm Friday November 11**. This design review consists of demonstrating your progress to date to a member of the course staff. More details about expectations and course staff office hours are available below.

## Learning Goals
1. Design and implement a larger scale program than previously.
2. Make good choices about which functions to design and their inputs and outputs.
3. Do more complicated things with strings, lists, and loops.
4. Read and write files.
5. Identify and fix errors.

## The Assignment
For this project, you will implement a simplified version of sentiment analysis, the process of determining whether a bit of text is positive, negative, or neutral. For ambiguous or unfamiliar words, the sentiment may not be clear. Consider the moment when Princess Leia calls Han a Nerf herder. Is that good? Looking at the rest of her statement gives us clues:

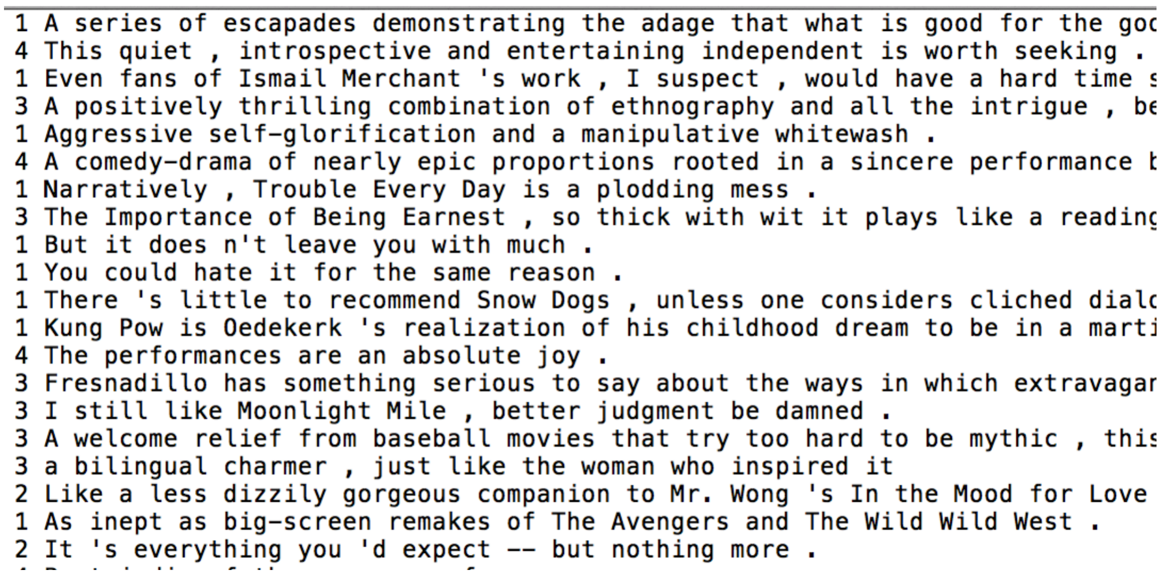
Why, you stuck up, half-witted, scruffy-looking… Nerf herder!
— Princess Leia, Star Wars Episode V: The Empire Strikes Back

Since we believe stuck up, half-witted, and scruffy-looking to be negative, we suspect that nerf herder is as well.

But how did we come to believe that stuck up, half-witted, and scruffy-looking were negative? Lots of examples of what was a compliment and what was an insult. This is one of the key principles behind machine learning – using lots of tagged examples to build models that can be used to make predictions about new items. In this way, your program will analyze a large set of examples that are tagged by sentiment to judge the sentiment of words in new input.

## Input data
In order to judge the sentiment of words, you are going to search through a training file containing movie reviews from the Rotten Tomatoes website which have both a numeric score as well as text. You may hard code this file name into your program. You'll use this to build a

simple model of which words are positive and which are negative. The training data file looks like this:



```
1 A series of escapades demonstrating the adage that what is good for the goc
4 This quiet , introspective and entertaining independent is worth seeking .
1 Even fans of Ismail Merchant 's work , I suspect , would have a hard time s
3 A positively thrilling combination of ethnography and all the intrigue , be
1 Aggressive self-glorification and a manipulative whitewash .
4 A comedy-drama of nearly epic proportions rooted in a sincere performance b
1 Narratively , Trouble Every Day is a plodding mess .
3 The Importance of Being Earnest , so thick with wit it plays like a reading
1 But it does n't leave you with much .
1 You could hate it for the same reason .
1 There 's little to recommend Snow Dogs , unless one considers cliched dialc
1 Kung Pow is Oedekerk 's realization of his childhood dream to be in a marti
4 The performances are an absolute joy .
3 Fresnadillo has something serious to say about the ways in which extravagar
3 I still like Moonlight Mile , better judgment be damned .
3 A welcome relief from baseball movies that try too hard to be mythic , this
3 a bilingual charmer , just like the woman who inspired it
2 Like a less dizzily gorgeous companion to Mr. Wong 's In the Mood for Love
1 As inept as big-screen remakes of The Avengers and The Wild Wild West .
2 It 's everything you 'd expect -- but nothing more .
```

Note that each review starts with a number 0 through 4 with the following meaning:

- 0 : negative
- 1 : somewhat negative
- 2 : neutral
- 3 : somewhat positive
- 4 : positive

**Basic Functionality**

Your program will perform a number of analysis operations, including

1. *calculating the sentiment score of a single word from the user:*
   you will ask the user to enter a word, and then you will search every movie review for that word. If you find it, add the score for that review to the word's running score total (i.e., an accumulator variable). You also will need to keep track of how many appearances the word made so that you can report the average score of reviews containing that word back to the user.
2. *calculating the average sentiment score of the words in an input file:*
   ask the user to give you the name of a file containing a series of words and compute the score of every word in the file. Think of the words in this file as either a compliment or insult that you are trying to judge the sentiment of. Report back to the user the average score of the words in the file. This will allow you to predict the overall sentiment of the phrase represented by words in the file. An insult has an average sentiment of less than 1.75, while a compliment has an average sentiment of more than 2.25.
3. *finding the highest and lowest scoring words in a file:*
   ask the user to give you the name of a file containing a series of words and compute the score of every word in the file. Report back to the user which word was the most positive and which was the most negative.

Your program should employ a simple menu that allows the user to pick the functionality that they want from the choices. When finished with it, present the menu again until the user chooses to exit. A sample run of the menu might look like this:

```
What would you like to do?
   1. Calculate the sentiment score of a single word
   2. Calculate the average score of words in a file
   3. Find the highest and lowest scoring words in a file.
   4. Exit the program
Enter a number 1-4: 1
Enter a word: heartbreaking
"Heartbreaking" appears 11 times
The average score for reviews containing "heartbreaking" is 3.18

What would you like to do?
   1. Get the sentiment score of a single word
   2. Get the average score of words in a file
   3. Find the highest and lowest scoring words in a file.
   4. Exit the program
Enter a number 1-4: 2
Enter the name of the file with the words: words.txt
The average score of the words in words.txt is 2.33. This is a compliment.

What would you like to do?
   1. Get the sentiment score of a single word
   2. Get the average score of words in a file
   3. Find the highest and lowest scoring words in a file.
   4. Exit the program
Enter a number 1-4: 3
Enter the name of the file with the words: words.txt
The most positive word in words.txt is puppy with a score of 4.23
The most negative word in words.txt is taxes with a score of 0.73

What would you like to do?
   1. Get the sentiment score of a single word
   2. Get the average score of words in a file
   3. Find the highest and lowest scoring words in a file.
   4. Exit the program
Enter a number 1-4: 4
```

**Advice and Suggestions**
Do not try to program the entire project up at once. You need to break this program down into simple pieces that you code and test one at a time.

Here is one possible way you could start this solution:

0. Use a "temporary" **main()** to interact and test each function as you create and write them.
1. Start with a smaller version of the data file to reduce your processing time.
2. Implement the menu first – have each option call a function that will initially just print a diagnostic message.
3. Get a word from the user and find all the lines in the data file that contain that word.
4. After that works, add the part that averages the scores together.
5. And so on….

**Some comments on coding standards**
For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable.

- You should be commenting your code, and using constants in your code (not magic numbers or strings).
- Any numbers other than 0 or 1 are magic numbers!
- Adhere to the coding standards by including function header comments for each of the functions (other than main).
- Including an explicit collaboration statement describing any people you have talked with about this project or any sources you have consulted (or an explicit statement that you collaborated with no one). A reminder: googling around to find information, ideas, or solution elements is strictly against course policy. Do not google at all in the course of working on this project.

**Extra credit**
Add capabilities to your program (and options to your menu). Be sure that your menu and prompts make usage clear to the user. Each of these additional capabilities are worth a possible five extra credit points:

1. Add a Mad-Libs function for generating comments. Read a comment from a user-supplied file with missing words indicated by an underscore ("_"). Replace each missing word with a word from the training file that is found with the same two surrounding words. For instance, if the comment was "My favorite show is The _ Place." and the training file contained the phrase "the right place", the modified comment might be "My favorite show is The right Place.". If there are multiple candidate words, pick one randomly. Be sure to match the original capitalization.
2. Increase the positivity or negativity of a comment in a user-supplied file by swapping out a single word with another word that has a more extreme (positive or negative) score. You should attempt to ensure that the new word makes sense in the context. One method might be to choose a new word that has a same word before and/or after as the word to be replaced. For instance, if the comment was "A strange combination of dull and suspenseful", you might increase the positivity by swapping out "strange" for "thrilling", a more positive word that can be found preceding "combination" in the training data file. For candidate words with

a matching surround, pick the one with the most extreme change (ie, the one with the most negative or positive sentiment).

**Preliminary design review**

The preliminary design review is worth 10% of this project. For the preliminary design review, you should write a description of the project design structure as you did in Lab 8. Specifically, you should sketch the contents of the program, breaking it down into **data** and **functions** and then order the functions into a program structure (basically an outline of the program). Your sketch should contain the following elements:

1. Data: what data is required by the project
   Variable names w/ description of its purpose
   Data type of each variable (string, int, list, etc.)

2. Algorithms:
   High-level processes the program needs to have.

3. Full Program pseudocode:

   - Arrange the various functions in order that they need to happen.
   - Include loops and some notion of when they terminate.
   - Place the necessary functions in the loops.
   - The pseudocode should contain a mix of high- and low-level items. Some aspects you might not solve until you start coding, so it's fine to give a high-level description.

Explain your project design structure to a member of the COS 125 staff during office hours or before/after your lab. Your design is due before 5pm on Friday November 11.

**How to turn in your homework**

Turn in each program in its own file. When turning in your own assignment make sure to add your last name to the file name (for example: Rheingans_p1.py). You should not turn in the training file. If you do the extra credit, submit two files: YourName_p1.py with the base assignment and YourName_p1-xc.py with the extra-credit version. If you do the extra credit, submit a few of your favorite Mad-Lib comment files.