YEAH Hours Assignment 6

Elina Thadhani

Warm Up: Practice with Dictionaries

Goal: analyze some data on disease infections at different locations. We are given a data file, where on each line we start with the name of a location, and then we have seven values (integers) that indicate the cumulative number of cases of a disease found at that location over the first seven days, respectively

```
Evermore , 1, 1, 1, 1, 1, 1, 1
Vanguard City,1 ,2 ,3 ,4 ,5 ,6 ,7
Excelsior   ,1,1, 2,   3,     5,      8,       13
```

Warm Up: Reading the File

Goal: Read in the file and return a dictionary in which the keys are the names of locations in the data file, and the value associated with each key is a list of the (integer) values presenting the cumulative number of infections at that location

- Tips:
  - Remember to take into account extra white spaces
  - When you read in a file, you are reading in strings but you want to store a list of integers
  - Super helpful function: **line.split(“”)** will split a line into a list by space
  - Super helpful function: **line.strip()** will remove extra white spaces

Warm Up: Calculating Infections per day

Goal: The function should return a new dictionary in which the keys are the same locations as in the dictionary passed in, but the value associated with each key is a list of the seven values (integers) presenting the number of new infections each day at that location.

For every day, except the first, you can determine the number of new cases by subtracting the cumulative number of cases on the day before from the cumulative number of cases on that day.

Note: return a **new** dictionary !

BiasBars

Two files: **biasbarsdata.py** for data processing and **biasbars.py** for data visualization

Important: you should not change any of the function names or parameter requirements that we already provide to you in the starter code.

BiasBars: Milestone 1

Goal: Write a program in **rating_stats.py** which loads the file **"data/full-data.txt"** and, using all the data in that file, calculates and prints two numbers:

1. The percent of reviews for women which are rated high (the rating is greater than 3.5)
2. The percent of reviews for men which are rated high (the rating is greater than 3.5

- **Ignore the first line of the files**
- **Each line:**
    - **1st value = numerical rating (1-5)**
    - **2nd value = gender (M or W)**
    - **3rd value = comment**

Calculate the percentage of reviews which are high for a given gender: divide the count of the number of reviews which are both for that gender *and* high by the count of the number of reviews for that gender. Then, to turn this decimal into a percent, first multiply by 100 and then convert to an integer using the **round()** function.

BiasBars: Milestone 2

In **biasbarsdata.py**

- < 2.5 = "low reviews".
- 2.5 and 3.5 (inclusive on both ends of range) = "medium reviews
- > 3.5 = "high reviews"

We are storing a dictionary of dictionaries where each entry in the outer dictionary is a word (key) to a dictionary with genders (key) and number of low, medium and high reviews (values)

```
{
    'great': {
        'W': [30, 100, 800],
        'M': [100, 200, 1500]
    },
    'teacher': {
        'W': [330, 170, 852],
        'M': [402, 250, 1194]
    }
}
```

BiasBars: Milestone 2

Add_data_for_word:

1. Find the word in your outer dictionary
2. Edit your inner dictionary

1. When you encounter a new word, add it!
2. Use convert_rating_to_index to help you determine what category (or index of the list) your rating falls into

add_data_for_word(word_data, "class", "W", 5.0)

Before:

```
{
    "class": {
        "W": [0, 0, 1],
        "M": [1, 0, 0]
    }
}
```

→

After:

```
{
    "class": {
        "W": [0, 0, 2],
        "M": [1, 0, 0]
    }
}
```

BiasBars: Milestone 3

Goal: process the whole file in **read_file**

Each line represents a single review and is composed of three comma-separated values. The first value is a numerical rating, the second value is a string describing the professor's gender, and the third value is a string representing the textual comment left as part of the review

You should process all the lines in the file

BiasBars: Milestone 4

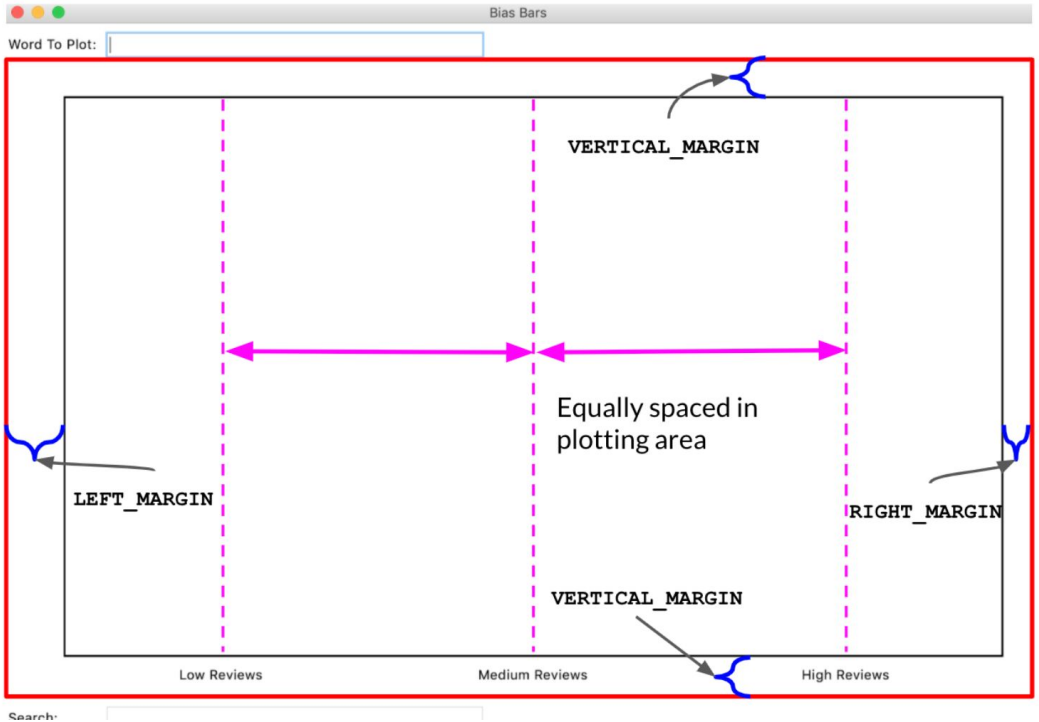
Goal: search for words in your dataset. Given a `word_data` dictionary and a target string, and return a list of all words from our dataset that contain the target string. This search should be case insensitive!!

This piece of code tests if string x is in string y:

If x in y:

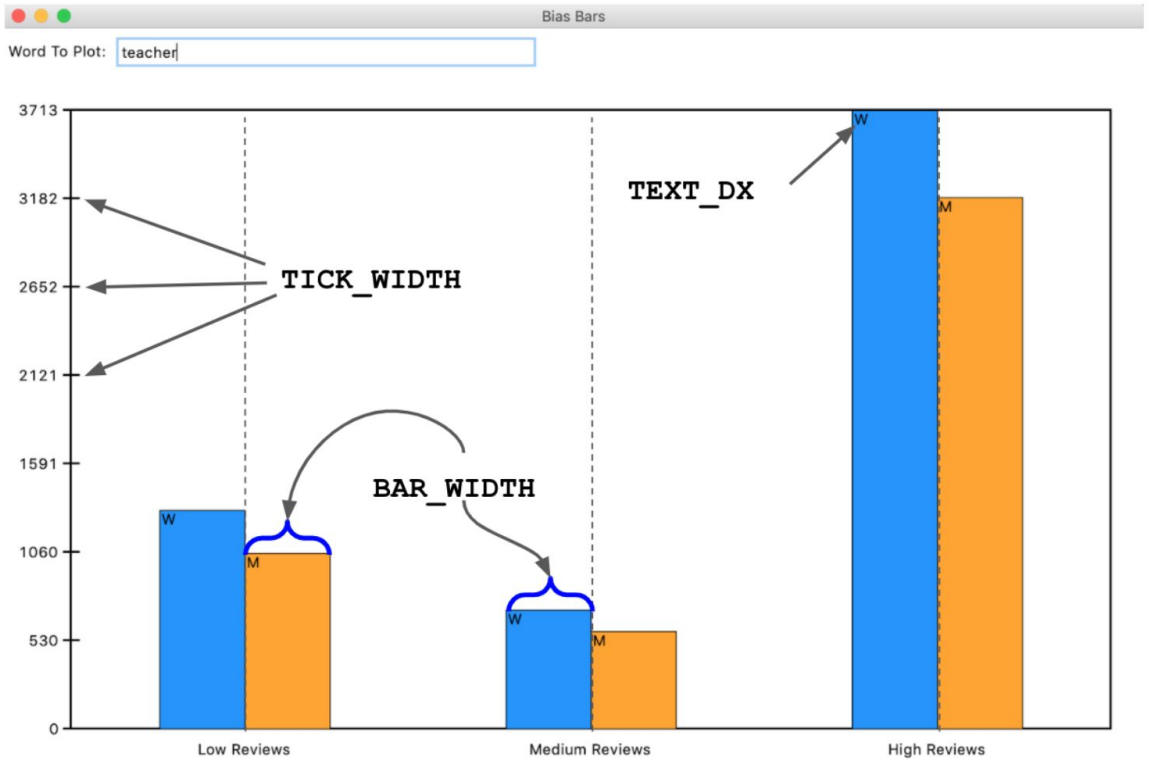
This works for substrings as well! Neat!

BiasBars: Milestone 6



Use constants!

BiasBars: Milestone 7



Think about how to get your x and y coordinates!