CSCI 1100 — Computer Science 1 Homework 8
Dictionaries and Classes

Homework Overview

This homework is made up of three parts. Each point is worth 40 points, but the "total" on the homework is only 100 points. Therefore, you can earn up to 20 bonus points toward your homework grade. For those of you intending to move on to Data Structures, Part 3 is quite important because it gives you more practice with classes. The assignment is due **Friday May 6, 2016 at 11:59:59 pm**. Please review the rules about excess collaboration from HW 3 to make sure you are turning in a program that is your own.

Please check your available late days on the homework submission server before planning your time on this assignment. Even if you have three days remaining, you can only use two on this assignment.

As always, good coding style is expected and low quality code will be penalized.

Part 1: Yelp Business Categories (40 points)

This first part revisits our Yelp exercise in a different way using a JSON formatted file. JSON was briefly covered in Lecture 13, but you should be able to figure it out based on the following. To begin, you are given the actual Yelp database from Lab 4 in JSON form in a file called `businesses.json`. Each line of the file is a JSON formatted string corresponding to a business. For example, the following program will read and print the some of the information for the business represented on the first line of the file:

```
import json
f = open('businesses.json')
line = f.readline()
business = json.loads(line)
print business['name']
print business['categories']
```

will result in the following output:

```
Mustang BBQ
[u'Food', u'Specialty Food', u'Meat Shops', u'Barbeque', u'Restaurants']
```

The function call `json.loads(line)` turns the JSON line into a dictionary with a number of entries. You can see these for this first business by adding a for loop to the above code to loop through the keys and print the key and the values. All of the keys are strings, but the values are booleans, integers, floats, lists and strings! For example, for the key `categories` the value is a list of "Unicode" strings — strings that can handle many different language character encodings. This is nothing that you need to worry about here because you can just treat them like regular strings.[1].

Write a program that reads a category name (correctly spelled) and a cutoff value (a valid integer). Your program must scan through the businesses — remember there is one business per line — and determine the counts for all categories that co-occur with the given category. It

[1]Note that Python may give you the same output without the u: ['Food', 'Specialty Food', 'Meat Shops', 'Barbeque', 'Restaurants'] This is perfectly valid and will not impact your program.

should then print those categories with counts greater than the given cutoff in sorted order. Right align the category names to 30 characters. (Remind yourself what the string function `rjust` does.) For example, after processing the above business, we will find that categories `'Food','Specialty Food','Meat Shops','Barbeque'` have occurred once. You will update the counts as you find more occurrences of a list containing the input category!

Here are some possible runs of your program:

```
Enter a category ==> Food
Food
Cutoff for displaying categories => 5
5
Categories co-occurring with Food:
                        Grocery: 5
     Ice Cream & Frozen Yogurt: 6
                   Restaurants: 8
```

```
Enter a category ==> Restaurants
Restaurants
Cutoff for displaying categories => 4
4
Categories co-occurring with Restaurants:
        American (Traditional): 4
            Breakfast & Brunch: 4
                       Chinese: 6
                         Delis: 4
                          Food: 8
                         Pizza: 17
                    Sandwiches: 4
                       Seafood: 4
```

Note: interestingly, not all `Restaurants` are considered `Food` and more restaurants are considered `Pizza` than `Food`! Here is one more example.

```
Enter a category ==> Shopping
Shopping
Cutoff for displaying categories => 2
2
Categories co-occurring with Shopping:
  Books, Mags, Music and Video: 3
                  Comic Books: 2
                Home & Garden: 2
                      Jewelry: 2
```

If the searched category is not found, your program must print `Searched category is not found.` If there are no categories to print (none co-occurring or none above the cutoff), your program must print `None above the cutoff`. See example runs below.

```
Enter a category ==> restaurants
restaurants
Cutoff for displaying categories => 4
4
Searched category is not found
```

```
Enter a category ==> Shopping
Shopping
Cutoff for displaying categories => 5
5
Categories co-occurring with Shopping:
None above the cutoff
```

Part 2: Yelp Business Reviews (40 points)

In this part, we will use two files. The first one is the file called `businesses.json` described in Part 1. Note that each business has a `'business_id'` that uniquely identifies that business. Two businesses may have the same name, corresponding to the different stores in different locations, but they will still have different `'business_id'` values. For example, in the file given to you, you have two different stores for `"Hannaford Supermarkets"` with `'business_id'` values: `'UZFBK4pU-VcEt1N4IoYCRA'` and `'LOOTfAyhOYR8fVuido_9aA'`.

In addition to this, you are given a second file called `reviews.json`. This file contains one review for a business in each line. You are only interested in the review text and the `'business_id'` for each review. For example, the following program will read and print the review for the first line of the file:

```
import json
f = open('reviews.json')
line = f.readline()
review = json.loads(line)
print review['business_id']
print
print review['text']
```

will result in the following output:

```
gekPL0Kc7w9zLDT3EulvxQ

Great fish fry.  The fish is fresh and comes in a great portion size.  Service is good.
    Having left the area, I really miss this place.
```

Note that the review will appear on a single line. The line break in this document is just an artifact of the line being too long to fit on the page.

Write a program that reads the name of a business using `raw_input` and finds **all** the reviews for **all** the stores of the given business. Your program must print the reviews in order they are found in the file, formatted with four spaces on the left and broken into lines using the `textwrap` module, which you'll have to figure out yourself. Use the default textwidth. Note: `textwrap` will remove newline characters. In this input data, paragraph breaks are indicated by two consecutive newlines. Make sure you print one blank line between paragraphs in a review. If the business requested is not found, print a message (see below). If no review is found for the given business, print a message (see below).

Example runs are given below.

```
Enter a business name => Bombers Burrito Bar
Bombers Burrito Bar
This business is not found
```

```
Enter a business name => Rensselaer Polytechnic Institute
Rensselaer Polytechnic Institute
No reviews for this business are found
```

```
Enter a business name => Country True Value Hardware
Country True Value Hardware
Review 1:
    Best customer service, and they have everything.

Review 2:
    This is my one stop shop for most of my basic hardware and home and
    garden needs because I'm in and out of here in no time. There's enough
    parking around the building, which is close to the front door. Country
    True Value is not the cheapest pricewise, but most items are
    convenient for me to find. The employees are easy to track down and
    are very helpful in locating items in the store when I need them.

    Earlier this month, I ended up buying my grass seed here instead of at
    Home Depot because I couldn't wait around for the Home Depot  employee
    to use the forklift while it was pouring rain outside. Of course, I
    paid a little more here for the convenience.

    Sometimes smaller is better!
```

Finally, note that the business below has multiple stores. You need to print the reviews for all locations:

```
Enter a business name => Ted's Fish Fry
Ted's Fish Fry
Review 1:
    Great fish fry.  The fish is fresh and comes in a great portion size.
    Service is good.  Having left the area, I really miss this place.

Review 2:
    Best fish fries in the Albany area.

Review 3:
    Ted's Fish Fry now has an official website and a Facebook page!
    Congrats on almost 60 years in business!

    They have some very cool photos on both, hehe :-O

    WEBSITE: http://www.tedsfishfry.net

    FACEBOOK: http://www.facebook.com/pages/Watervliet-NY/Teds-Fish-
    Fry/240983270362
```

```
Review 4:
    3 stars is just about right. The fish frys are very good, although I
    didn't like their chili sauce...it was more like relish, but there
    were a lot of regular customers ordering it so ymmv. Had a fish fry
    with cocktail sauce and tartar that was very good. The staff here just
    yell the food back to the kitchen help and they are always getting it
    screwed up or miscommunicated...comical, maybe that is part of the
    charm? I also had a hot dog and the sauce is sort of bland...I prefer
    hot dog sauce with a lot more flavor and kick. They also had carrot
    cake and cheesecake available...figured they had to be great...the
    carrot cake was a little dry. Go eat fish frys, figure out what you
    like on them and you will be good to go.
```

Part 3: Battleship Game (40 points)

In this part, you will implement a simple game of battleship. The list of all battleships (or ships as we will call them) and all player moves are given in a single file. Your program must read the name of the file using `raw_input`.

The first line of the file tells you how many ships there are, followed by the description of each ship, then, all the player moves. Read the ships first, then perform all player moves until either no more moves are left or a player wins.

Your program must define and use a class called `Battleship` that stores relevant information about each ship. Define the class in the same file as your homework description to simplify submission. Each ship has the following information:

`Name|x|y|length|height|health`

Each ship is a rectangle with upper left corner at `x,y` and lower right corner at `x+length, y+height`. The health defines how many hits it takes to sink this ship. Good news: ships don't move. Even better news: all players take aim at the same set of ships for simplicity.

A player move is given by:

`Player Name|x|y`

which means that a player with the given name has fired a torpedo at the coordinates `(x,y)`. You can assume players are taking turns and the input is valid. Each time a player hits a ship, display a message. If the player misses completely, display a message as well. Anyime the ship is hit, its health is decremented by one and when the health reaches zero, the ship has sunk — Display a message. A player cannot hit a sunken ship. When printing ship names, right justify them 12 characters.

Anytime a player sinks the last ship, the game ends immediately and that player wins. Print a message and stop your program. If all the moves are played and there are still some ships left, your program must print `No player won!`.

We have provided two sample inputs and outputs in the ZIP file for this homework.

Submission Instructions

Please note that the files we use on the submission server are likely going to be different than the ones we have given you.

Please follow these instructions carefully.

Part 1. Submit a single file called **hw8_part1.py** that assumes the existence of a file called **businesses.json**. Your program must read a single category and an integer cutoff value using `raw_input` and return all the matching categories in sorted order.

Part 2. Submit a single file called **hw8_part2.py** that assumes the existence of two files called **businesses.json** and **reviews.json**. Your program must read the name of a restaurant using `raw_input` and return all the matching reviews in the order they are found in the file.

Part 3. Submit a single file called **hw8_part3.py** that reads the name of a file using `raw_input` containing the number of ships, the information for each ship on a single line and then all the player moves. Items on a line are separated by `|`.