

CSCI 1100 — Computer Science 1 Homework 9

Dictionaries and Classes

Homework Overview

This homework is made up of three parts, each part is **40 points**. Part 1 is required. Part 2 and 3 are optional/bonus. If you complete all three parts, you can make up to 80 points lost in past homeworks. Each part is relatively short, so you have plenty of time to try them all. All three parts are due **Thursday December 10, 2015 at 11:59:59 pm**. All submission rules, academic integrity rules, and late penalties apply. Please pay careful attention to the submission instructions at the end of this document for each part. Note that these are all short and straightforward programs, my solutions including empty lines for formatting have 33, 33 and 60 lines.

Part 1: Yelp Business Categories

You are given the actual Yelp database from Lab 5 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 any print the business information from 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']
```

Associated with each business is a list of categories given by the Yelp users. Recall that JSON will return Unicode strings (e.g. `u'Food'` above) which you can treat like regular strings.

Write a program that reads a category name (correctly spelled) and a cutoff value (a valid integer). Your program must count all categories that co-occur with the given category and print those categories with counts greater than the given cutoff in sorted order. Right align the category names to 30 characters. For example, after processing the above business, we will find that categories `'Food'`, `'Specialty Food'`, `'Meat Shops'`, `'Barbeque'` each occur at least 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 algorithm:

```
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 are above 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 (optional/bonus part)

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 is associated with a '**business_id**' that uniquely identifies that business. Two businesses may have the same name, corresponding to the different stores in different locations. These will 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 'L00TfAyhOYR8fVuido_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 any 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 (formatting added by me):

```
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.
```

Write a program that reads the name of a business using `raw_input` and finds 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 `textwrap`. Note: If there are new lines in the review, you must preserve those (textwrap will remove them). If the searched business is not found, print a message. If no review is found for the given business, print a message.

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 is 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:

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 :-0

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 (optional/bonus part)

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 are given. Read the ships first, then perform all player moves until either no more moves 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. Best news: ships don't move. Even better news: all players take aim at the same set of ships for simplicity.

Each 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 hits completely, display a message as well. Anytime the ship is hit, its health is decremented by one. Whenever the health reaches zero, the ship has sunk. We display a message. A player cannot hit a sunken ship. When printing ship names, right justify them 12 characters.

Anytime a player sinks a ship and there are no ships left, the game ends immediately and display that the current player has won. 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 `hw9_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 `hw9_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 `hw9_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 `|`.