CSCI 1100 — Computer Science 1 Homework 8
Dictionaries

Homework Overview

This homework is worth **110 points** toward your overall homework grade and is due **Saturday November 15, 2014 at 11:59:59 pm** to give you extra time to try the extra credit portions. But, you must remember to give yourself sufficient time to study for the exam as well, which will on monday November 17. This homework also consists of one single program. Please review the rules about excess collaboration from HW 3 to make sure you are turning in a program that is your own.

You are expected to learn how to use dictionaries for this homework. If you do not use dictionaries, you will loose points.

As always, you are expected to match your output to the one given on the submission server. Your grade is going to depend almost completely on code correctness and structure, very little on the actual formating of the output (see below).

What you are expected to do in this homework is very simple: implement a better version of Yelp! That is all.

Be careful: This is a long homework. The longest one you will be given in this class. Give yourself ample amount of time to write it and test it. You are given extra time because you need it. Do not procastinate.

As much as the solution is long, it can be made better by better design and use of functions. My solution for the homework is 67 lines (including some blank lines for readability) and 81 lines for the extra credit version. Make use of the full power of all the constructs given to you: lists, dictionaries and sets. Sets can be super useful for the extra credit. Finally, you do not have to use functions but I *highly* recommend that you use some. Think about making your code more readable and easier to debug with the help of functions. It is common practice to keep the main code (below the `''__main__''` line) very simple and push all remaining processing to functions.

Yelp Data

You are given a file called `businesses.json` containing information from Yelp about businesses in the Rensselaer region, a longer version of the file you worked with in Lab #4. Each line of the file is a JSON formatted string for a single business. You can convert these strings to a Python object using the module called `json`.

To get this started, first try the following on WingIDE:

```
import json
for line in open('businesses.json'):
    b = json.loads(line)
    print b
    break
```

We are looking at the first line of the file (hence the break). By executing this, you would get the dictionary below:

```
{'city': 'Cohoes',
 'review_count': 2,
 'name': "Joe's Tavern",
 'neighborhoods': [],
 'url': 'http://www.yelp.com/biz/joes-tavern-cohoes',
 'type': 'business',
 'business_id': 'A_Fm4v2-gQuGBBI-GBx4Uw',
 'full_address': '16 Division St\nCohoes, NY 12047',
 'latitude': 42.7808878,
 'state': 'NY',
 'longitude': -73.7102039,
 'stars': 4.0,
 'schools': ['Rensselaer Polytechnic Institute'],
 'open': True,
 'categories': ['Pizza', 'Restaurants'],
 'photo_url': 'http://s3-media1.ak.yelpcdn.com/assets/2/www/img/924a6444ca6c/gfx/blank_biz_m
```

Let us go through this step by step.

Structure of information for a business

First, take a look at the dictionary for a business. The variable `b` is of type dictionary. It has 16 keys, of these `name` is a string value, `full_address` is a string, `categories` is a list of strings, `review_count` is an integer, and `stars` is a float. You will also need the `latitude` and `longitude` of the business (both floats) for distance computation. The remaining fields for a business will not be needed for this homework.

JSON

We used a new module called Json, which is used to convert values between built-in Python objects and strings. Json is a frequently used data format in many Web sites, stands for JavaScript Object Notation.

You can convert a Python object to string using the function `dumps()`, and you can convert a string containing a valid json format to a Python object using `loads()`. You can play with this to see the results:

```
>>> import json
>>> x = {'a':[1,2], 'b':True}
>>> y = json.dumps(x)
>>> y
'{"a": [1, 2], "b": true}'
>>> json.loads(y)
{u'a': [1, 2], u'b': True}
```

Note that `json` does not recognize sets, so you cannot store sets in a string using this module. The act of converting an object to a string is called *serializing* it.

Unicode

As in the above example, when you decode a line from the file we gave you, you actually see something like:

```
{ u'city': u'Cohoes',
  u'review_count': 2,
  ...
}
```

where each string has a preceding `u`. This means that all the strings are encoded as Unicode. For all intensive purposes, the strings in Unicode are just regular strings. You can just use all the string functions that you use as before. So, don't worry about the `u` and skip the rest of this part.

If you want to know more about Unicode, read on. Unicode encoded strings are the same as regular (ASCII) strings if you only use characters from the English alphabet.

```
>>> u'cat' == 'cat'
True
```

When you use special characters that are not in English, the internal code may vary depending on the encoding:

```
>>> x = u"Schrödinger's cat"
>>> print x
Schrödinger's cat
>>> x
u"Schr\xf6dinger's cat"
>>> y = "Schrödinger's cat"
>>> print y
Schrödinger's cat
>>> y
"Schr\xc3\xb6dinger's cat"
```

Internal encoding method is not necessarily Unicode and hence can have different code for the same letter. This will also differ depending on specific settings in your computer. If you use Unicode, it will always be the same. This is not really important to solving this homework, but you must understand why Unicode is uniformly used in programming text when you cannot force people to use only English spelling of words. (*PS. I had to tell my editor for this homework to use Unicode to print the above example as well!*)

Homework Requirements

You should write a program that allows the user to:

- Continuously ask the user first for a location. Location is required.
- If the user enters an empty location or `-1`, your program should stop its main loop and exit. You should not ask for a category in this case.
- If a location is entered, you must ask for a category to filter by. Category is optional. If the user leaves the category blank, then it means there is no category criteria.
- If a location is provided, your program should find all businesses at this given location according to the `full_address` field, anywhere in the address string and within the input category if a category is input by the user.
- Report to the user:
  1. the number of businesses matching the given criteria
  2. the max distance between all the businesses that match the given criteria (see the explanation below for the distance computation)
  3. the categories the matched businesses fall under and the number of matched businesses within each category (dictionary is very useful here).
- Ask the user whether she wants to see the list of businesses.
  - If the answer enters anything but `Y` or `y`, go back to asking for location and category.
  - If the answer is `Y` or `y`, print the name, number of reviews, the number of stars and the address for each matching business.
- **Note 1:** If 0 or 1 businesses are found, the max distance between them is clearly 0. A note of advice: you cannot use the `max()` function on an empty list.
- **Note 2:** You will ask whether to list the businesses only if you find at least 1 matching business.
- **Note 3:** There is no ordering to the printed businesses. I print them in the order they appear in the file. You can choose whatever is convenient to you. The categories are ordered alphabetically which I expect you to do as well for readability.

Your program should work regardless of how the information is formatted, lower case, upper case or any type of capitalization.

Remember: try to get as close to this as possible, but if you do not we will give plenty of partial credit. In fact, I describe the approximate rubric so that you can see what each part is worth.

Finding maximum distance between businesses

To find the distance between two points, you will use a function given to you in `hw8util` module which takes in two latitude and longitude values (`dist(lat1,long1,lat2,long2)`), and returns a distance in miles.

For example, we can find the distance between RPI Union and Dinosaur BBQ (in as the crow flies type distance):

```
>>> import hw8util
>>> hw8util.dist(42.730863399999997,-73.681679299999999,42.734591000000002,-73.688817)
0.4445660592440111
```

To find the maximum distance between a number of businesses, you must first find all possible pairwise distances and then take the max. For example, if we have three businesses, `a,b,c` with distances:

```
distance between a and b is 3 miles
distance between b and c is 0.2 miles
distance between a and c is 2.8 miles
```

then you must return 3 miles as the maximum distance. Try to write your program so that you only compare any pair of values only once! This will become very important as you work with larger and larger set of values.

Extra Credit (20 points)

Homework extra credit allows you to make up for lost homework points. If you have more than 100 in the homework average, we will truncate it at 100. But, you must still attempt this for an extra challenge.

Change your program so that the user can also enter multiple categories separated by commas (there may be spaces that you must strip). Each category must be in the results. Furthermore, if you entered ∼ before a category, than that category must not be in the results returned. For example:

`pizza, italian` returns businesses that have both `pizza` and `italian` in their categories.

`pizza,~italian` returns all businesses that have `pizza` but `not italian` in their category.

`pizza, ~italian, food` returns all businesses that have `pizza` and `food` but `not italian` in their category.

As before, if no category is given, then your program should give all businesses in the given location as before. The remainder of your program works the same way.

Deliverables – Final Check

- You must use the program structure outlined in Homework # 7, i.e.

  ```
  import json

  def function1(x):
      return x + 1

  if __name__ == '__main__':
      z = 10
      print function1(z)
  ```

  With no global variables above the `if __name__ == '__main__':` line.

- Your program must use dictionaries and the `json` module to read the file. After that you can do what you want. But using a dictionary will simplify your code considerably.

- Submit a single file called `hw8.py` that assumes the existence of a text file called `businesses.json`. The test file will be identical to the one we give you by the way.

- The main loop of the program should be exit if -1 or nothing is entered for location, otherwise it should loop and ask for the next location and category. Otherwise, you will get EOF errors.

- Print any input that you read as we have been doing.

- Here are some (rough) grade guidelines:

  10 points: correct program structure, 20 points: correct execution of your program while loop and interaction with the user (we cannot test your code without this), 30 points: finding the businesses correctly, 20 points: printing the matching categories and counts, 10 points: printing the max distance between found locations, 20 points: printing matching business info.

  I recommend building your program in this order, and testing slowly.

NOTE: We will have a final test case for the extra credit. If you do not implement the extra credit, your program should return nothing on this part.

Expected output

Below you can see the expected functioning of this program (I put some dividers for easy readability, you are not required to match this):

```
==============================
Where (-1 to exit) ==> Wynantskill
Wynantskill
Filter by what ==>

        American (Traditional) (1)
        Beauty and Spas (1)
        Beer, Wine & Spirits (1)
        Burgers (1)
        Convenience Stores (1)
        Fast Food (1)
        Food (4)
        Grocery (1)
        Ice Cream & Frozen Yogurt (1)
        Italian (2)
        Japanese (1)
        Nail Salons (1)
        Pizza (2)
        Restaurants (7)
Found 12 businesses in 1.68 radius
Print the businesses (y/Y)? ==> n
n

==============================
Where (-1 to exit) ==> troy
troy
Filter by what ==> no category
no category
Found 0 businesses, with 0 radius

==============================
Where (-1 to exit) ==> troy
troy
Filter by what ==> American (traditional)
American (traditional)
        Active Life (1)
        American (Traditional) (15)
        Bars (1)
        Burgers (2)
        Diners (2)
        Event Planning & Services (1)
        Food (1)
        Food Stands (1)
        Golf (1)
        Hot Dogs (1)
        Ice Cream & Frozen Yogurt (1)
        Nightlife (1)
        Pizza (1)
        Restaurants (15)
        Seafood (1)
        Sports Bars (1)
        Steakhouses (1)
        Venues & Event Spaces (1)
Found 15 businesses in 7.58 radius
Print the businesses (y/Y)? ==> y
y
1. Manory's Restaurant (13 reviews, 3.0 stars) [99 Congress St Troy, NY 12180]
2. Dakota Restaurant (7 reviews, 2.5 stars) [579 Troy Schenectady Rd Latham, NY 12110]
3. Brunswick Greens Clubhouse & Restaurant (3 reviews, 4.5 stars) [1004 Hoosick Rd Troy, NY
4. Happy Lunch (2 reviews, 3.5 stars) [827 River St Troy, NY 12180]
5. South End Tavern (12 reviews, 3.5 stars) [757 Burden Ave Troy, NY 12180]
6. Red Front Restrnt & Tavern (20 reviews, 4.0 stars) [71 Division St Troy, NY 12180]
7. Nighthawk's Kitchen (12 reviews, 5.0 stars) [Riverfront Park Troy, NY 12180]
8. The Brown Bag (22 reviews, 4.5 stars) [156 4th St Troy, NY 12180]
9. MJ's on the Avenue (4 reviews, 2.5 stars) [499 2nd Ave Troy, NY 12182]
10. Holmes & Watson (10 reviews, 3.5 stars) [450 Broadway Troy, NY 12180]
11. Friendly Restaurants (4 reviews, 2.5 stars) [120 Hoosick Street Troy, NY 12180]
12. Forty One Sports Bar & Grill (3 reviews, 5.0 stars) [41-112th Ave Troy, NY 12182]
13. Famous Lunch (42 reviews, 4.0 stars) [111 Congress St Troy, NY 12180]
14. The Eatery (6 reviews, 4.0 stars) [452 Bloomingrove Dr Troy, NY 12180]
15. Country View Diner (21 reviews, 3.0 stars) [855 Hoosick Rd Troy, NY 12180]

==============================
Where (-1 to exit) ==> -1
-1
```

Here is a sample for the **extra credit** (you will have 0 matches if you did not implement this part).

```
==============================
Where (-1 to exit) ==> troy
troy
Filter by what ==> pizza, italian
pizza, italian
        Italian (3)
        Pizza (3)
        Restaurants (3)
Found 3 businesses in 4.37 radius
Print the businesses (y/Y)? ==> y
y
1. Testo's Restaurant & Pizza (14 reviews, 4.5 stars) [853 4th Ave Troy, NY 12182]
2. Notty Pine Tavern (6 reviews, 4.5 stars) [2301 15th St Troy, NY 12180]
3. DeFazio's Pizzeria (53 reviews, 4.5 stars) [266 4th St Troy, NY 12180]

==============================
Where (-1 to exit) ==> troy
troy
Filter by what ==> italian, ~pizza
italian, ~pizza
        Delis (1)
        Italian (6)
        Restaurants (6)
Found 6 businesses in 4.89 radius
Print the businesses (y/Y)? ==> n
n

==============================
Where (-1 to exit) ==>
```