Computer Science 1 — CSci 1100
Lab 4 — Yelp Data
Fall Semester 2014

Lab Overview

This lab explores use of lists and logic for analyzing real data provided by Yelp for restaurants near Rensselaer. You will use a module that will help you with parsing files. We will learn about files very soon, but feel free to look at the code and ask about what it is doing during the lab.

In order to get started, please create a folder in your Dropbox for Lab 4 and download the zip file `yelp.zip` from the Piazza Resources page into this new folder. Unzip it to find two files, `lab04_util.py` and `yelp.txt`. The first is a module for reading the second file. First, take a look at the file: `lab04_util.py`. You can see it has functions for parsing the file, but no code to call these functions. That is a simple way to think of a module. We can use these files in our code to simplify some tasks.

Checkpoint 1

Let's use the given module to read this file into a list. Create a new file called `check1.py` in the same folder as the files from the zip folder and include the following code inside:

```
import lab04_util

restaurants = lab04_util.read_yelp('yelp.txt')
```

Use a few print statements to see the contents of the list. As the list is large, let's look at the first element:

`print restaurants[0]`

We will get:

```
["Meka's Lounge", 42.74, -73.69, \
 '407 River Street+Troy, NY 12180',\
 'http://www.yelp.com/biz/mekas-lounge-troy', \
 'Bars', \
 [5, 2, 4, 4, 3, 4, 5]]
```

So:

- The variable `restaurants` contains a list.
- Each element of the list is a specific restaurant.

  The information for any restaurant as it appears in the list is: name, latitude and longitude, street address, URL, the type of restaurant and a list of scores given by Yelp users.

- The last elements of a restaurant is another list, with multiple scores given by Yelp users.

Your job in the first checkpoint to print information for a single restaurant with the help of a function `print_info`. Here is the result of printing the first two restaurants in the list:

```
Meka's Lounge (Bars)
        407 River Street
        Troy, NY 12180
Average Score: 3.86

Tosca Grille (American (New))
        200 Broadway
        Troy, NY 12180
Average Score: 2.50
```

First line is name and the category in a paranthesis. The second and third line both come from the address after a TAB. The final line is the average score, obtained by taking the average of the last entry in the restaurant.

How do we split the address into two lines? Well, there is a handy function called `split()` that splits a string into a list based on a given delimiter. For example:

```
>>> title = "My,name,is,red"
>>> title.split(",")
['My', 'name', 'is', 'red']

>>> title = "My|name|is|red"
>>> title.split("|")
['My', 'name', 'is', 'red']
```

Just to get you started, here is a basic organization for printing just the name of a restaurant.

```
##############################
import lab04_util

def print_info(restaurant):
    print restaurant[0]

####### main code starts here
restaurants = lab04_util.read_yelp('yelp.txt')
print_info(restaurants[0])
##############################
```

To complete Checkpoint 1, show the lab TA or a mentor the code and the output.

Checkpoint 2

Before getting started, copy `check1.py` to `check2.py` and continue to work on `check2.py`. Modify your code to ask the user for the id of a restaurant between 1 and 155 (humans don't need to know about list ids starting at 0). Assume the user enters a number. If the user enters a value outside of the range 1-155, print a warning and do nothing else.

Now, if the user has entered a valid index, you will print the information for the restaurant corresponding to this index (remember: index 1 corresponds to list index 0). Test your code well first to make sure that you only print a restaurant for a valid index.

The second task in this part is to improve on the print function by changing the average score computation. Given the scores for a list, you will drop the max and the min, and find the average of the rest. You have already done this for Homework #1 and you know that you do not actually have to explicitly remove the max, min, just subtract them from the sum.

Compute average of the remaining scores and instead of printing the average score, print one of the following based on the score:

Score	Output
0 up to 2	This restaurant is rated bad, based on x reviews.
2 up to 3	This restaurant is rated average, based on x reviews.
3 up to 4	This restaurant is rated above average, based on x reviews.
4 up to 5	This restaurant is rated very good, based on x reviews.

where x is the real number of the reviews for this restaurant.

Beware: it does not make sense to remove max and min if there are less than three reviews for a restaurant. In that case, we should use average of all the values (another if statement!).

To complete Checkpoint 2, show the lab TA or a mentor the code and the output. Please check to make sure your code follows the structure we require: first imports, then functions and then the actual code. Test your code with values 8, 22, 33, and 44.

Checkpoint 3

Before getting started, copy `check2.py` to `check3.py` and continue to work on `check3.py`. We will add a final flair in this part to your program from part 2.

Your program should work exactly as it did in part 2. After printing the restaurant info, you will now ask the user following:

```
What would you like to do next?
1. Visit the homepage
2. Show on Google Maps
3. Show directions to this restaurant
Your choice (1-3)? ==>
```

For all these options, using formatted strings will really simplify your life!

If the user answers `1`, then pop up the browser using the following command, but using the URL for the business instead of this address. Remember to import module `webbrowser` first of course.

```
webbrowser.open('http://xkcd.com/1319/')
```

If the user answers `2`, then pop up the browser with Google maps with the address of the business.

```
webbrowser.open('http://www.google.com/maps/place/business-address-goes-here')
```

If the user answers `3`, then pop up the browser with Google maps with the address of the business and Renssealer. Here is an call:

```
webbrowser.open('http://www.google.com/maps/dir/business-address/rpi-address')
```

If the user answers anything else, your program does nothing.

For example, to find the location of Rensselaer, you can use the following call:

```
webbrowser.open('http://www.google.com/maps/place/110 8th Street Troy NY 12180')
```

Luckily, Google can handle spaces or even pluses in an address. So you do not need to really work too hard to change the address string.

To complete Checkpoint 3, show the lab TA or a mentor the code and the output.

Note. This lab had lots of different components. As a result, it can really benefit from structuring the code in an easy way so that you can quickly modify and improve it. Take the time to show your code to your TAs and mentors, work with them in restructuring it. Get lots of advice. This will be crucial in the future when we write (even) longer code. It helps to design functions to do simple things only: print restaurants just prints info at a certain index. Think of of all input output as part of your main program.