Computer Science 1 — CSci 1100
Lab 5 — Yelp Data
Fall Semester 2017

Lab Overview

This lab explores the use of lists and logic to analyze real data provided by Yelp for restaurants near Rensselaer. You will use a module that we provide to help you parse a file of data. It contains file functions that we have not yet covered in class, but don't worry, we will cover this material very soon. In the meantime, feel free to look at the code and ask about what it is doing during the lab.

To get started, create a folder in your Dropbox for Lab 5 and download the zip file `yelp.zip` from the Piazza Resources page into this new folder. Unzip it to extract two files, `lab05_util.py` and `yelp.txt`. The first is a module for reading the second file. Use the Wing IDE to take a look at `lab05_util.py`. You can see it has functions for parsing the `yelp.txt` file, but no code to actually call these functions.

Checkpoint 1

Let's use the `lab05_util.py` module to read `yelp.txt` into a list and do some simple operations on it. Create a new file called `check1.py` in the same folder as the files from the zip folder and include the following code inside:

```python
import lab05_util

restaurants = lab05_util.read_yelp('yelp.txt')
```

Use a few print function calls to test the file read and see the contents of the list. For example, to see the first element, use: `print(restaurants[0])`, you 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]]
```

Printing other elements will give you similar information for a different restaurant in the list. So:

- The variable `restaurants` contains a list.
- Each element of the list is information on a specific restaurant, also organized as a list. This restaurant information contains (in order): name, latitude and longitude, street address, URL, the type of restaurant and a list of scores given by Yelp users.
- Importantly, the last element of the list of restaurant information is another list, with multiple scores given by Yelp users.

Your job in the first checkpoint to develop a `print_info` function to print information for a single restaurant. 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
```

The first line in the output is the name of the restaurant, followed by its restaurant type in parentheses. The second and third line are the street address and city/state/zip code components of the address. Both are printed after first printing a TAB character. The final line is the average score, obtained by taking the average of the last entry in the restaurant. You may assume there is at least one score, and you must output the average score accurate to two decimal places.

The one unknown is how do we split the address into two lines? Think about our handy friend `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']
```

You can use the split function on the restaurant address to get the pieces you need.

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

```
import lab05_util

def print_info(restaurant):
    print(restaurant[0])

####### main code starts here
restaurants = lab05_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 for restaurant 0 and restaurant 4.

Checkpoint 2

Before getting started, copy `check1.py` to `check2.py` and develop your code for `Checkpoint 2` in `check2.py`.

Now modify your code to ask the user for the id of a restaurant from 1 up to and including the length of the `restaurants` list (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 value range, print a warning and do nothing else.

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

The second task in this part is to improve on the print function by changing the average score computation. Given the scores for a restaurant, you will drop the max and the min, and find the average of the rest. Remember that you do not have to explicitly remove the max, and min from the list, just subtract their values from the sum.

Once you have computed the average, instead of printing it, 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 total number of 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 this case, we should use the 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 your work in `check3.py`. In this part we will add a final flair to your program from part 2 using the module `webbrowser`. Remember to import it at the beginning of your program,

Your program should start exactly as it did in part 2, but after printing the restaurant info, you will now ask the user the 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)? ==>
```

Using formatted or multi-line strings here will really simplify your life!

If the user answers `1`, then your program should pop up the browser using the following command, but using the URL for the business instead of this address.

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

If the user answers `2`, you will need to show the location of the restaurant. The call to show a location on Google Maps is:

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

It can be used to show RPI:

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

If the user answers 3, you should pop up the browser with Google Maps directions from RPI to the given business. You can use the following format:

```
webbrowser.open('http://www.google.com/maps/dir/from_address/to_address')
```

Here is an example for directions between RPI and Troy Farmers Market:

```
webbrowser.open('http://www.google.com/maps/dir/110 8th Street Troy NY/49 4th St
    Troy, NY 12181')
```

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

Google can handle spaces or even pluses in an address. So you do not need to work too hard to change the address string for options 2 or 3.

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, and work with them in restructuring it. Get lots of advice. The ability to appropriately structure your code will be crucial in the future when we write (even) longer code. It helps to design functions to do simple things only: `print_info` just prints info at a certain index. Think of of all input and output as part of your main program.

Extra work to challenge yourself

If you are done but want to continue improving this program, here are a few ideas. There are no bonus points for these, but they are great exercises and you will learn by doing them.

We have just learned about loops in Lecture 9. The first thing to try is to print all of the businesses by looping over the indices of the list. This should be a nice simple extension to your program.

The second thing to try is to ask the user to enter a business name. Now, you need to go through each of the businesses with a loop just like you did above. However, if the business name matches the one you just entered, then you should call your function to print the information. If you get through the loop without finding a match, then you should print that the business is not found.

You can expect similar problems in HW#4, so this is a good time to practice.