Principles of Urban Informatics
Assignment 1

Posted on: 09/08/2014
Due Date: 09/15/2014

Data description

In this assignment we are going to build a set of python functions to process 311 complaints data. In all the problems we are going to use a subset of the 311 data obtained at `https://nycopendata.socrata.com/`.

The data is given in the following format

```
Unique Key,Created Date,Closed Date,Agency,Agency Name,Complaint Type,
Descriptor,Incident Zip,Status
27241168,01/24/2014 10:43:00 AM,01/24/2014 01:00:00 PM,DEP,Department
of Environmental Protection,Asbestos,Asbestos Complaint (B1),11222,
Closed
27039760,01/02/2014 12:00:00 AM,01/05/2014 12:00:00 AM,HPD,Department
of Housing Preservation and Development,PLUMBING,BASIN/SINK,11211,
Closed
27289992,01/28/2014 11:57:27 PM,,DOB,Department of Buildings,Building/
Use,No Certificate Of Occupancy/Illegal/Contrary To CO,11219,Open
27228583,01/23/2014 09:09:00 AM,01/23/2014 12:00:00 PM,DSNY,A - Staten
Island,Snow,E9 Snow / Icy Sidewalk,10314,Closed
...
```

Problem 1

Given an input data file, your script should find the range of dates in which complaints were created. The code should get the name of the dataset as a command line parameter and output the result as follows:

```
> python problem1.py sample_data_problem_1.csv
10 complaints between 01/02/2014 00:00:00 and 01/28/2014 23:57:27
```

Note: dates should be formatted exactly as shown, i.e., month/day/year hour:minutes:seconds):

Problem 2

Given an input data file, display the number of complaints of each complaint type (case sensitive), and output as in the example below:

```
> python problem2.py sample_data_problem_2.csv
Building/Use with 1 complaints
Asbestos with 1 complaints
APPLIANCE with 1 complaints
Non-Residential Heat with 1 complaints
Street Light Condition with 1 complaints
Snow with 1 complaints
HEATING with 2 complaints
GENERAL CONSTRUCTION with 1 complaints
PLUMBING with 1 complaints
```

Problem 3

Similarly to problem 2, process the given input data file to output the number of complaints for each complaint type, but this time ordered in descending number of complaints, i.e., from the types with most complaints to fewer as in the example:

```
> python problem3.py sample_data_problem_3.csv
HEATING with 2 complaints
APPLIANCE with 1 complaints
Asbestos with 1 complaints
Building/Use with 1 complaints
GENERAL CONSTRUCTION with 1 complaints
Non-Residential Heat with 1 complaints
PLUMBING with 1 complaints
Snow with 1 complaints
Street Light Condition with 1 complaints
```

Note: if that happens, sort alphabetically the complaint types that have the same number of complaints, as in the example.

Problem 4

Given an input data file and an integer k, your script should compute and output the top-k complaint types in terms of number of complaints. The code should get the name of the dataset and k as command line parameters, and output the result ordered in descending number of complaints as follows:

```
> python problem4.py sample_data_problem_4.csv 3
HEATING with 2 complaints
APPLIANCE with 1 complaints
Asbestos with 1 complaints
```

Example of how to handle draws: if there are 5 top complaint types (say B, A, C, E, D) with the same number of complaints, a top-3 query should show only types A, B, C as below:

```
> python problem4.py sample_data_problem_4.csv 3
A with 221 complaints
B with 221 complaints
C with 221 complaints
```

Note: if that happens, sort alphabetically the complaint types that have the same number of complaints, as in the example.

Problem 5

Given an input data file, your script should compute the number of complaints per day of week. The code should get the name of the dataset as a command line parameter and output the result as follows (should start on Monday and end on Sunday):

```
> python problem5.py sample_data_problem_5.csv
Monday == 0
Tuesday == 2
Wednesday == 1
Thursday == 2
Friday == 4
Saturday == 0
Sunday == 1
```

Note: we recommend to use the *strptime* from the time package to extract the day of the week of a date.

Problem 6

Given an input data file, output for each agency the zip code that generates the largest number of complaints for that agency. The agencies should be listed listed in alphabetical (only output agencies with at least one complaint that has a valid zipcode). You should not assume all complaints contain a valid zip code and agency. The code should get the name of the dataset as a command line parameter and output the result as follows:

```
> python problem6.py sample_data_problem_6.csv
DEP 11222 1
DOB 11219 1
DOHMH 11220 1
DOT 11214 1
DSNY 10314 1
HPD 10011 10459 10460 11209 11211 1
```

In each line of the output, the first string is the agency name, the second one corresponds to the zip code (or zipcodes, see below for handling ties) with the maximum number of complaints for the agency and the last number corresponds to the number of complaints in the selected zip code for the agency in question.

Note: if two or more zip codes are tied as with most complaints, output the list of all zipcodes with the maximum number of counts in lexicographical order, as in the example above (the agency HPD has multiple zipcodes with 1 complaint in this data file).

Problem 7

Now, use the zip_borough.csv to count the number of complaints per borough (and also using the previous data file). The zip_borough.csv contains a list of zip codes and, for each zip code, its borough.

The code should get the name of the dataset and the zip-borought.csv files as a command line parameter and output the result as follows:

```
> python problem7.py sample_data_problem_7.csv zip_borough.csv
Brooklyn with 112 complaints
Queens with 66 complaints
Bronx with 65 complaints
Manhattan with 44 complaints
Staten Island with 13 complaints
```

You can assume that all complaints in the dataset contain a valid zip code and borough. Note that the sum of all complaints in the output is 300, equal to the number of complaints in the input; this means that all complaints in the input file sample_data_problem_7.csv have a valid zip.

Problem 8

Repeat exercise 7, but do not assume that all complaints have valid zip codes, i.e., some lines contain a blank zip code and should be ignored. The input and output should follow the same pattern as Problem 7. For example:

```
> python problem8.py sample_data_problem_8.csv zip_borough.csv
Brooklyn with 98 complaints
Bronx with 62 complaints
Manhattan with 51 complaints
Queens with 50 complaints
Staten Island with 16 complaints
```

Note that the sum of all complaints in the output is not 300. This means that some of the complaints do not have a valid zip and have been ignored.

Setting up

If you still have not python installed in your system, you can check the instructions for installation in the Dive into Python book available at `http://www.diveintopython.net/`. In this course, we will be using Python 2.* (e.g., 2.7 should be enough). If you install any version 3.* the syntax is going to be a little different.

The sample datasets can be obtained at `http://vgc.poly.edu/projects/gx5003-fall2014/week1/lab/data/sample_data_problem_x.csv`. Similarly the zip_borough file can be obtained at `http://vgc.poly.edu/projects/gx5003-fall2014/week1/lab/data/zip_borough.csv`.

Questions

Any questions should be sent to the teaching staff (Instructor Role and Teaching Assistant Role) through the NYU Classes system.

How to submit your assignment?

Your assignment should be submitted using the NYU Classes system. You should submit all your python code (do not submit the data files) in a zip file named NetID_assignment_1.zip, you should change NetID, by your NYU Net ID. As illustrated above, for each problem you should create a *.py* file called problemx.py, where x should be the problem number.

Grading

The grading is going to be done by a series of tests and manual inspection when required. Notice that grading by manual inspection is very subjective and therefore, you should try as much as possible make your code run as specified so that the amount of subjective grading is minimized. Make sure that your code runs on the sample datasets as specified.