

Principles of Urban Informatics

Assignment 2

Posted on: 09/15/2014
Due Date: 09/22/2014

Data description

In this assignment we are going to implement a set of functions to store and manage job positions data. In all the problems we are going to use a subset of the NYC Jobs data obtained at <https://data.cityofnewyork.us/Business/NYC-Jobs/kpav-sd4t>.

The job positions contain the following attributes (the order of the attributes is used through the assignment):

Job ID|Agency|# Of Positions|Business Title|Civil Service Title|Salary Range From|Salary Range To|Salary Frequency|Work Location|Division/Work Unit|Job Description|Minimum Qual Requirements|Preferred Skills|Additional Information|Posting Date

In this list, Job ID is a unique key generated for each job position and therefore the same value of this attribute *can not* appear more than once in the database.

A sample csv file with job offers can be found at:

http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/src/NYC_Jobs_sample.csv.

A sample csv file with job offers can be found at:

http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/NYC_Jobs_sample.csv.

Your task is to write a python program that receives a set of commands and performs actions on a file-based database. Your database should be persistent, which means that even after your program ends, the data should be stored in *text* files for future use. You should try to use the concepts discussed in class. See <https://docs.python.org/2/tutorial/inputoutput.html> for a tutorial on how to write files in python.

Commands and Input Files

The input files are going to be given as a sequence of commands (see list below) one command per line. In the following we describe the set of commands that your program must support:

- *clear*
removes all the data in the database.
- *insert|field_value1|field_value2|field_value3|...*
adds a job offer into the database. As shown, it receives a set of field values in the same order as in the example above, separated by “|”. If an attempt is made to insert a job position with a Job ID that is already in the database, the insertion should not happen.
- *delete_all|field_name|field_value*
deletes from the database all job offers for which *field_name* = *field_value*.
- *update_all|query_field_name|query_field_value|update_field_name|update_field_value*
changes the value of the attribute *update_field_name* to *update_field_value* for all job offers for which *query_field_name* = *query_field_value*.
- *find|field_name|field_value*
outputs all the job offers for which *field_name* = *field_value*. The attributes should be printed in the same order as in the example above, and the job offers should be sorted by the job Id.
- *count|field_name|field_value*
outputs the number of job offers for which *field_name* = *field_value*.
- *dump*
writes all job offers to a file called *dump.txt*. Columns should be in the same order as in the example above, and jobs should be sorted by job Id.
- *view field_name1|field_name2|field_name3|...*
prints a *view* of the job offers with only the attributes *field_name1|field_name2|field_name3|...*. Columns should appear in the order specified in the view, and job offers should be sorted by job Id.

Problem 0

Describe your strategy to store the database? Which files are you going to generate? Why? Try to relate to the concepts presented in class.

Problem 1

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_1.txt. They are equivalent to:

- *clear*
- *insertl...*
- *insertl...*
- ...
- *insertl...*
- *dump*

This series of commands will clear the db and insert several jobs, possibly with different IDs, and finally dump all job offers in the database. A line trying to insert a job with an existing job Id should be ignored, and dump should output all job offers sorted by job Id, with the columns defined in section *Data description*, as can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_1.txt. Your job is to implement a *insert()* method so that the output is exactly as in the example.

Problem 2

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_2.txt. They are equivalent to:

- *clear*
- *insertl...*
- ...
- *insertl...*
- *update_all\query_field_name\query_field_value\update_field_name\update_field_value*
- ...
- *update_all\query_field_name\query_field_value\update_field_name\update_field_value*
- *dump*

This series of commands will clear the db and insert several jobs. After that, it will execute a series of updates: all job offers that match the criteria *query_field_name=query_field_value* will have its field *update_field_name* changed to *update_field_value*, and print the number of job offers that were updated (possibly 0 when no job offer matches the criteria). The output of the series of commands in the example can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_2.txt. Your job is to implement a *update_all()* method so that the output is exactly as in the example.

Problem 3

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_3.txt.

They are similar to:

- *clear*
- *insertl...*
- *...*
- *insertl...*
- *delete_all|id|120851*
- *dump*

This series of commands will clear the db and insert several jobs, with different IDs. After that, one job will be deleted from the db. You have to implement the command *delete_all* such that it receives a *field_name* (in this example id) and a *field_value* (in this example 120851), and deletes all jobs with *field_name* equal to *field_value*. In this example input, the job with id 120851 will be deleted.

Your job is to implement a *delete_all()* method so that the output is exactly as in the example.

The output of this series of commands can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_3.txt.

Problem 4

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_4.txt.

They are similar to:

- *clear*
- *insertl...*
- *...*
- *insertl...*
- *delete_all|agency|DEPARTMENT OF FINANCE*
- *dump*

Notice that we are no longer deleting the job by its ID. The command *delete_all* must also support this type of deletes. The command must delete all jobs such that the *field_name* (in this example agency) is equal to *field_value* (in this example DEPARTMENT OF FINANCE). In other words, you have to delete the agency DEPARTMENT OF FINANCE and all the jobs associated with it.

Your job is to modify your *delete_all()* method so that the output is exactly as in the example.

The output of this series of commands can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_4.txt.

Problem 5

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_5.txt. They are equivalent to:

- *clear*
- *insertl...*
- ...
- *view\field_name1\field_name2l...*

This series of commands will clear the db, insert several jobs, with different IDs. After that, we are creating a view with specific columns. The output of this series of commands can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_5.txt. Your job is to implement a *view()* method so that the output is exactly as in the example, i.e., only the selected columns are shown (in the specified order), and job offers are ordered by job offer Id.

Problem 6

Consider the commands in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_data_problem_6.txt. They are equivalent to:

- *clear*
- *insertl...*
- ...
- *find\field_name\field_value*
- ...
- *find\field_name\field_value*

This series of commands will clear the db, insert several jobs, with different IDs. After that, we are searching for job offers per specific field names and field values, and outputting nothing when no job offer matches the criteria, or the full job offers which have `field_name=field_value`. The output of this series of commands can be seen in http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_output_problem_6.txt. Your job is to implement a *find()* method so that the output is exactly as in the example, i.e., when searching for an inexistent `field_value`, print nothing! otherwise print the job offers that match the find criteria, with the columns in the same order as detailed in section *Data description*, and ordering the job offers by job offer Id.

Setting up

If you have not done yet, refer to assignment 1 for instructions on how to have Python in your system.

You are given a sample code for you to use (it is optional). This code contains a possible structure for the code as well as hints for you to implement the required functionalities. You can find the sample code here http://vgc.poly.edu/projects/gx5003-fall2014/week2/lab/data/sample_code.py

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) and a text file with your answer to Problem 0 in a zip file named `NetID_assignment_2.zip`, you should change `NetID`, by your NYU Net ID.

Grading

The grading is going to be done by a series of tests and manual inspection when required. Make sure your code runs on the sample datasets as specified to minimize the need for manual inspection of the code, which can be very subjective.