CS 2316

Pair Homework 8 – Data Merge

Due: Friday, March 16th, before 11:55 PM
Out of 100 points

Files to submit: **1. HW8.py**

<u>This is a PAIR PROGRAMMING Assignment: Work with your partner!</u>
For pair programming assignments, you and your partner should turn in identical assignments. List both partners names at the top. Your submission must not be substantially similar to another pairs' submission. Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:
- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:
- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute** to do this assignment in case you run into problems.

Introduction:

The city of Hopeulikit posts public tax information about various properties at the following URL:
http://www.cc.gatech.edu/classes/AY2012/cs2316_spring/hw/hopeulikit-030112.html

You have been hired by a mortgage company to write a program that downloads the tax amount for each property to update their database. The mortgage company will provide your program with a .csv file containing data about all of the properties they know about (**properties-#####.csv**). [Where the ###### is a unique number specific to that version of the file.]

They want you to match each property in their file up with the corresponding property on the city website, and import the data (dollar amount, as a float) from the "tax rate" field. After you get this data, you should export it to a CSV file called **taxData-######.csv**. [Where the ###### matches the number of the input file they gave you.] Note that all records in the properties-####.csv file should be exported back out to the taxData-

######.csv file, even ones for which a tax rate is not found (leave that field blank).

Difficulties:
The city website does not present addresses nicely separated out into their component elements in the same format the mortgage company uses for it's database of properties. You will have to write code to extract the specific fields that match the mortgage companies fields yourself.

Sometimes clerks at the city make mistakes, and do not correctly key in all of the data. For example, sometimes they may transpose or miss a digit in street/house numbers or zip codes, or omit a “North” or “South” street direction, or use an abbreviation (“Rd vs Road”) for the street type or even omit the street type altogether. Clerks at the mortgage company may also make similar mistakes, so the data from the website and your input CSV file may not exactly match. Typically enough of the data matches so that you should be able to match the properties up, but you can't trust that all properties will match 100% perfectly! For the **taxData-######.csv** file, the mortgage company wants you to keep all of their provided property data from the **properties-######.csv** file and only add the “tax rate” field to it at the end of the record. (Regardless of what changes the city website may have for address of the property.)

Also, the mortgage company does not know about all properties in town (some properties are paid off, and have no mortgages) so the city website will have some properties that do not exist in the input .CSV data file. In addition, the city does not update property listings until the year after the properties are built, so the mortgage company may have records in their input .CSV file that do not yet have a tax bill (and won't even appear in the city website.) In short, some properties appear in only one of the two data sources, so you obviously can not match them up. The mortgage company needs to have a human examine these “non-matched” properties. Any properties that do not have a tax rate found on the city website should be left with the tax rate field blank in the taxData-###### file so that the morgage company can investigate them. Any properties found on the city website that do not match a record in the properites-######.csv data file will need to be exported into a file: **non-matched-city-######.csv** The mortgage company wants the formats of this non-matched-city file to match that of the provided properties-######.csv file, so you will have to extract the street number, street direction, street name, street type, city, zip and state from the city website and populate each field. Sort the non-matched properties by their street number, followed by street direction, street name, street type, etc... Use TEXTUAL sorting, not numeric sorting. (Sorted examples in order are “100” , “12”, “1327”,”566”, etc...)

If your matching algorithm fails, you may inadvertently miss a tax rate or save properties that have a match in the non-matched-city file. The mortgage company is basing your pay partially on how infrequent these matching mistakes are made, so try and match properties up well to minimize accidents and improve your bonus!

The mortgage company is hiring you to build a program it's managers can run, but they also want the source code! They specifies that the source code must be easy to read and

well documented (with comments) so that they can hire another student after you graduate to maintain the program and potentially add extra features.

Smaller test files:

Download the small_test.zip file to find a group of test files that include sample solutions. You will want to test on the smaller files first, and only after you get things working successfully move on to the larger files. The html file in this zip file is also posted here: http://www.cc.gatech.edu/classes/AY2012/cs2316_spring/hw/hopeulikit-030612.html

Larger test files:

Download the large_test.zip file to find a group of test files that are the same size as actual data set, but that include sample solutions. The html file in this zip file is also posted here:
http://www.cc.gatech.edu/classes/AY2012/cs2316_spring/hw/hopeulikit-031312.html

Table Web Scraper:

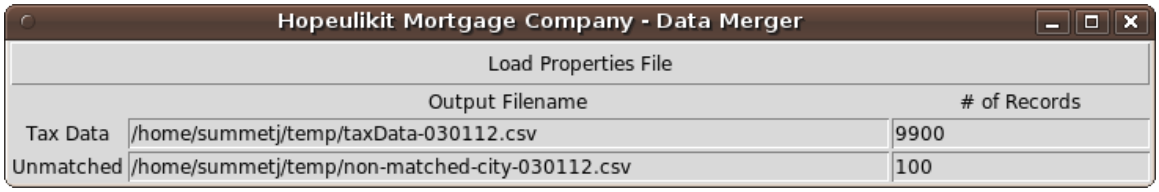
You will need a function or method that will retrieve the table of data at the above URL and convert it into a list of data records.

You may make use of the HTML table parser demonstrated in class and posted on the class course-notes webpage:
http://www.summet.com/dmsi/html/readingTheWeb.html#parsing-html-tables

(readTable.py), but you must indicate the source of that function with comments in your code. If you modify that function, be sure to use comments to explain how/why you modified the code so the TA's will know what extra work you have done.

Informational GUI

Write a simple GUI that has a button to load the properties-#######.csv file ("Load Properties File"). When the user clicks the button, the GUI will call your code to load the file. Then you should download the current data from the city website, match all the properties, and write out the various output files. Be sure to write the output files to the same directory as where the properties-######.csv file was loaded from (Even if that is not the current working directory!). Your GUI should display the full path and file name for each output file, along with a list of the number of records in that file. (Note that the taxData-######.csv file should have the same number of records as the input properties-######.csv fle, while the non-matched-city and non-matched-properties files should have many fewer records.) When your client (or the TA) runs your python file, your GUI should appear automatically. The following screenshot is illustrative only. Your GUI may look different as long as it provides all the same data.



Suggested Problem Solution

The following is a suggested way to solve this problem. You are encouraged to lay out your code using the function/method names and arguments given below. If your functions are in a class (methods) they will all need a self parameter, which is not shown.

__init__ or initializeGUI

This function/method will set up the GUI. Have a button for the user to select the properties-######.csv file via a file dialog. It should have a text entry box for the name and complete path of each output file (taxData and non-matched-city) that are read-only, as well as text entry fields which list the number of records saved in each file.

clicked

When the user clicks the button on your GUI, this method will do several things. First, it should ask the user for the filename. It should try and load the properties-######.csv file using *loadPropertyFile.* If it has an error loading the file, it should tell the user, and then stop. If the properties file loads successfully, it will then attempt to download the data from the city website using *downloadCityData.* If the download fails for some reason, it will tell the user and stop. If the download was successful, it will call *mergeData.* Finally, it will call *saveTaxData* and *saveNonMatchedData* to save the output data files. If either of those saves fail, it will notify the user and stop. *(We recommend that saveTaxData and saveNonMatchedData directly update the GUI with the saved filenames and number of entries, but you could also do this within the clicked method here.)*

downloadCityData

Optional Parameters: You can either hard-code the URL of the city website to download, or pass it in as a parameter to this function/method.
Output: This function/method should either return a list of the city data, or populate an object variable (such as self.cityData) that contains the downloaded data.

This method will go to the appropriate URL and download the data. You may want to use the parseTable.py file on the DMSI course-notes to help you parse the HTML table.

convertCityToMortgageFormat

This helper function will take in a record in the city format (two fields, one for the full address and one for the tax rate) and return a record in the format used by the mortgage company (a separate field for street number, street direction, street name, street type, city, state, zip, and taxRate). You may find regular expressions to be useful for this function, but their use is not required. This function could be called from *saveNonMatchedData*, but it is possible you may also find it useful when comparing city data to the mortgage company data and calculating the matching score between a mortgage company record and a city website record.

loadPropertyFile

This function/method will load all of the data from the properites-######.csv file. If the file does not exist, or does not have valid data, you should indicate that an error occurred, either by returning False or raising an exception. The method that called this function/method (*clicked*) will issue an appropriate error message to the user.

mergeData

This method/function will take each record from the mortgage companies data file, and try and find the best match from the city website. It does this by calling *calculateMatchScore* between the mortgage company record and every city property record, keeping track of which record gives the highest match score. Note that if NO property gives a “high enough” match score, you may have a property that just does not exist in the city website. You should give these properties a “blank” tax-rate field. (You could use an empty string to represent a blank field, but should NOT use a zero.) Once you have processed all properties you will eventually save this data using the *saveTaxData* method.

When you match a property from the mortgage company file to the city website, you should remove that property from the city website data structure so that you do not match against it accidentally a second time. Once you have processed all of the mortgage company records, any left-over records in the city properties data structure are by definition “non-matched” city properties. Later you will save these using the *saveNonMatchedData* method.

calculateMatchScore

This method will help your mergeData method by calculating a 'match score” between a record from the properites-######.csv data file and the list of records downloaded from the city website. This match score will be used to determine which property from the city website best matches a property from the mortgage company data file. How you calculate this score is entirely up to you. We are presenting one possible way below, but feel free

to use any method which gives you the desired results (of having the highest match score between the correct properties.)
One way to calculate a match score is to look at each question below and add up the points to come up with a total match score:

- Does the target address contain the street name? (5 points)
- Does the target address contain the street number? (3 points)
- Does the target address contain the same street direction? (1 point)
- Does the target address contain the same street type? (2 points)
- Does the target address contain the city? (5 point)

- Does the target address contain the state (5 points)
- Does the target address contain the zip code? (5 points)

saveTaxData -and- saveNonMatchedData

These two methods do almost the same thing, just with different data. They each save data to a file whose format is exactly like that of the input properties-######.csv file (plus one extra data field at the end of each record containing the tax rate as a float, or blank if the city website didn't have a tax rate for that property). The saveNonMatchedData method is slightly different because it has to convert the data from the city website format (two fields) to the format used by the mortgage company (a separate field for street number, street direction, street name, street type, city, state, zip, and taxRate). We suggest you implement a convertCityToMortgageFormat method to help you do this for each record.
You may also want these methods to update the GUI after they write the file to display the full path to the file and the number of records saved in each.

Grading

You will earn points as follows for each element that works correctly according to the specifications.

Basic Functionality:	**(70 points)**
Code is easily readable and understandable with good comments	**10**
Successfully loads the properties-######.csv file	**5**
Successfully downloads the webpage HTML	**5**
Correctly parses the HTML table to extract records	**10**
Correctly extracts specific fields needed from the city address	**10**
Output files contain correct data, are saved in correct directory	**10**
Information about the output files is displayed correctly in the GUI	**10**
Correctly matches the tax rate with the property > 50% of cases	**10**
Matching Accuracy:	**(Max 30 points)**
-Correct "tax rate" for at least 80% of the properties on city website	**10**
-or-	
-Correct "tax rate" for at least 95% of the properties on the city website	**15**
-NM city file is smaller than it should be	**0**
-NM city file is no more than 50 % larger than it should be	**5**
-or-	
-NM city file is no more than 25% larger than it should be	**10**
-or-	
-NM city file is no more than 5% larger than it should be	**15**

+5 Bonus EC points are available for matching ALL of the properties correctly and correctly identifying all of the non-matching properties. (Basically, if your output exactly matches the "perfect" solution.)