For our third practice project, your task is to join the cancer-risk data set that you loaded and processed in third course in this specialization to the county center data set that you generated from the map of the USA last week. In the final week of this specialization, you will then use this joined data set to plot cancer-risk data on the USA map. While this task is relatively straightforward, two important topics related to scripting are illustrated in this project: understanding the efficiency of your script and verifying the results of your script.

Practice Project: Reconciling Cancer-Risk Data with the USA Map

This practice project will consist of three main parts as described below.

Merge two county-level data sets by common FIPS code

Before writing any code, we first briefly consider how to approach this problem. Our provided code includes a function **`read_csv_file(f)`** that reads a CSV file and returns a list whose items correspond to rows in the CSV file. After reading in both data sets as tables, our task reduces to generating a third tables that represents the join data set. One simple approach is to iterate through the cancer-risk data set and check whether the FIPS code for a county also appears in the county center data set. If so, we generate a joined row in the output table. Following this approach, the key task is to determine whether a particular FIPS code appears in the county center data set. Our suggestion solution to this problem is to create a Python dictionary corresponding to the county center data set where the keys for the dictionary are FIPS codes and the values are the corresponding county centers.

To this end, your task for this part of the project is to write a function **`make_dict(table, key_col)`** that takes a list **`table`** and an integer **`key_col`** and returns a dictionary whose key/value pairs correspond to the rows in **`table`**. In particular, each key should correspond to the entry in column **`key_col`** while the corresponding value should be the rest of the row with the key deleted. Once you have implemented this function, you are welcome to test it using our provided function **`test_make_dict()`**.

The advantage of our suggestion approach is two-fold. First, the Python code required to join the two data set is simple once the function **`make_dict()`** is available. Second, using a dictionary to test whether a FIPS code is in the county center data set improves the speed of the resulting Python code significantly. In particular, Python can determine whether a key in dictionary in time **independent of the size** of the dictionary. (Python uses hash tables to implement dictionaries.) This week's programming tip discusses this topic in more detail.

Write a script that reads both CSV files, merges by FIPS code, outputs to a CSV file

For the second part of this project, your task is to write a Python function **`merge_csv_files(cancer_csv_file, center_csv_file, joined_csv_file)`** that takes the named cancer-risk and county center CSV files, read the files, merges them using the shared FIPS code, and writes the result to the named CSV file with the following columns:

- State
- County name
- FIPS code
- Population
- Cancer risk
- Horizontal coordinate of county center (with respect to the USA SVG map)
- Vertical coordinate of county center (with respect to the USA SVG map)

Once your have implemented this function, your task is to run this script on the trimmed cancer-risk CSV file from week 3 of the previous course and the county center CSV file from last week. To facilitate testing your script, we suggest that you create smaller versions of these two CSV files and run your code on these files first. Once you are confident in the correctness of your implementation, you are welcome to compare our version of the joined CSV file to your own result.

Investigate discrepancies in the two sets of FIPS code

Our last task for this project is to investigate the output of our Python script for merging the two data sets. One point of note is that building appropriate tests for `merge_csv_files()` is difficult. While testing your code on small hand-made example files is a good start, determining whether the output CSV file is truly correct is quite tricky. One check will be plotting the cancer-risk data at the computed county centers and visually verifying that the results are plausible (as done in the project for the final week).

Another practice that we highly recommend when working with large data sets for which the correctness of the computed answer is unclear is to be **extremely** vigilant concerning anomalies that exist in the data sets. Basically, if something seems strange about your data set, make sure that you understand the nature of the anomaly. This anomaly could just an issue with the data set or if could be a problem with your script. Your job as an expert scripter is to determine which is the case.

For the two data sets at hand, we note that the size of the two data sets (and the computed result) do not agree. In particular, the cancer-risk data set has 3276 rows while the county center data set has 3143 rows. The joined table computer in our solution has 3140. This anomaly leads to the obvious question of "Why/how do the sets of FIPS codes disagree?". For the final part of the project, your task is to add code to `merge_csv_files()` that identifies FIPS codes that appear in one data set, but not the other.

Once you have complete your analysis of the anomalous FIPS code, add an explanation for this anomaly to the bottom of your code as comments. When you have complete all three parts of this project, you are welcome to compare your code/comments to our solution.

Mark as completed