Lab Week 10 - Dictionaries

Skills Needed to complete this Lab

|  |
| --- |
| * Use dictionaries * Previous skills reading data from a file. * Read a file with the csv module |

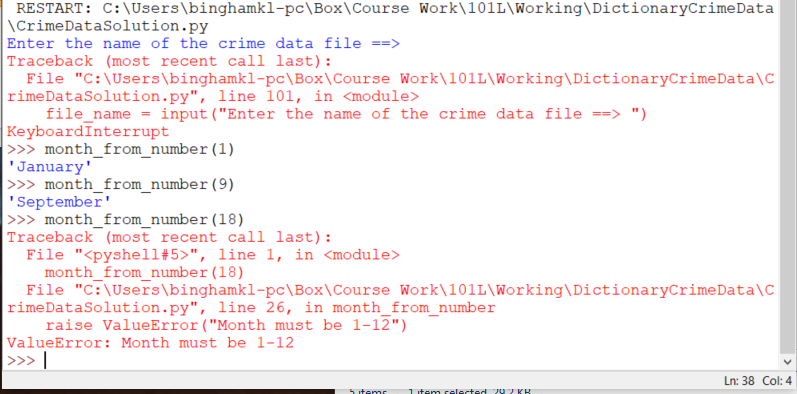
## Kansas City Police Crime Data

This week you’ll read in a datafile containing crime information for 2019. You will need to create the functions given below with the function signature given. Function signature means the definition of the name of the function as well as any parameters and return values. There will again be a unit test file to assist you. Each one will also show an example in the shell. Once you write a function you can just hit F5 and then call the function from the shell to test it with different values.

You can open the csv files in excel to view it as rows and columns, but make sure you don’t save any changes. You can also open the file in a text editor to see how it is actually stored character by character.

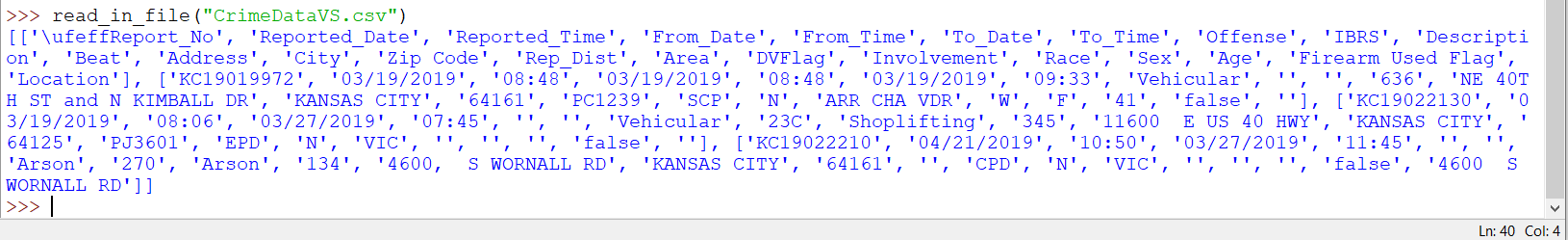
## month\_from\_number

The month\_from\_number function takes an integer as a parameter and returns a string. The parameter is expected to be 1 to 12 and returns the string result for the month. The list of months is January, February, March, April, May, June, July, August, September, October, November, December



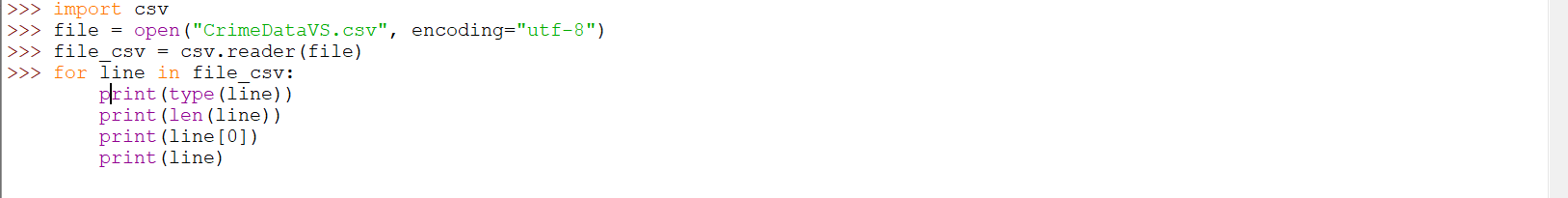
## read\_in\_file function

The read\_in\_file function takes a filename as a string and returns a list of list of list of the contents. Each sublist is a row from the file. The file is expected to be a comma separated value. You’ve been given 3 files with comma separated values.



Python has a built-in module to help you read csv files. It will automatically split by the commas as you read through the file.

You can try the following lines of code ( Make sure you the IDLE shell is in the same directory as your csv or you will get a FileNotFoundError )



Don’t forget to close the file. Try the example above. Each iteration in the for loop reads the line from the file\_csv variable as a list of strings. Each element being a comma separated value.

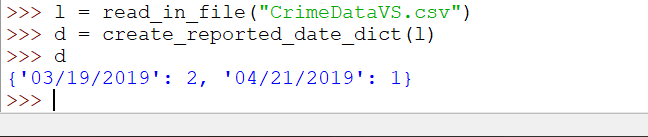
Note above that the file is opened with encoding=”utf-8”, make sure you open the file that way in the function as well. Also, many will be prompted to import csv just before using it. It’s better to place it once at the top of the module.

For this assignment, you’ll specifically want to know the reported\_date which is index 1, the offense at index 7, and the zip code at index 13. Notice in the shell example above the list of list returned from read\_in\_file has a row for the header.

Hint : Since line in the above sample of code is a list of the items you need, you could just append that into the list of lines that your function should return.

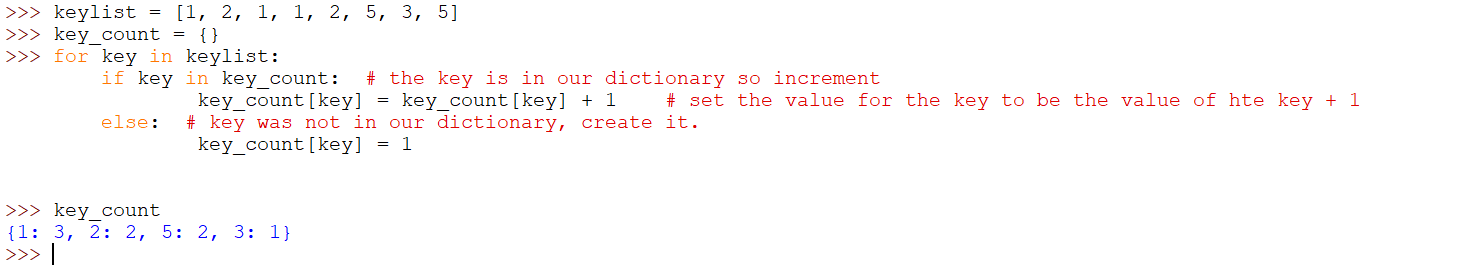
## create\_reported\_date\_dict function

The create\_reported\_date\_dict function takes a list, which is the list of lists returned from the read\_in\_file function above and returns a dictionary where the key is a date of the year found in index 1, and the value is how many times a crime occurred on that data as read from the file

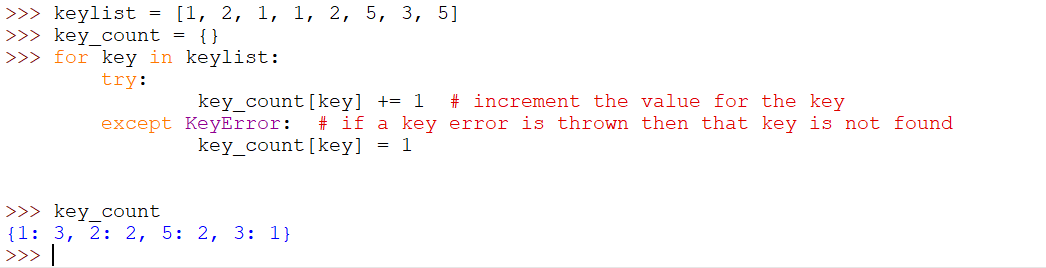


As you read through the list of lists, there is a couple of ways to update a value for a key for something like this. As you iterate over the list passed into the function; read in the date. If the date is already a key then you can increment the value, if it doesn’t, then this is the first time you’ve encountered this date and you can create the key value pair with a value of 1. There are 3 ways to do this as shown in the dictionary tutorial given on canvas. Below here is a sample.

Test to see if key exists in the dictionary

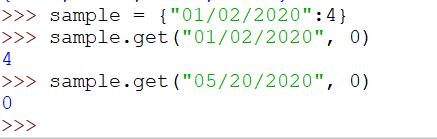


Referencing a key that doesn’t exist causes an error, so we can catch that exception

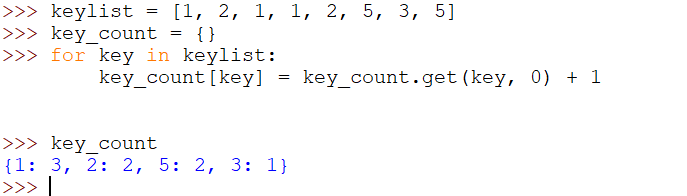


Use the get method of dictionaries

The get method takes one argument and an optional argument. The first argument is the key to check for, the second is the default value for the key. If the key exists it returns the value for the key, if it doesn’t it returns the default value for the key you’ve passed in.

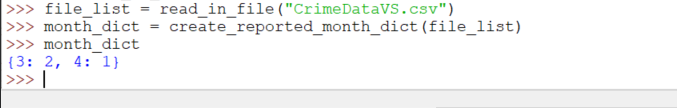


Get example



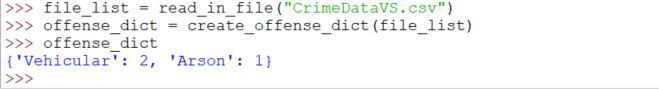
## create\_reported\_month\_dict function

The create\_reported\_month\_dict function takes a list, which is the list of lists returned from the read\_in\_file function above and returns a dictionary where the key is the month of the offense, and the value is how many times a crime occurred on that data as read from the file.



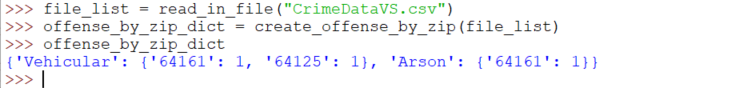
## create\_offense\_dict function

This function takes a list, again it is the list returned from read\_in\_file function and returns a dictionary where the key the offense (Arson, Burglary, etc) and the value is how many times that offense occurs. Offense is column index 7.



## create\_offense\_by\_zip function

This function takes a list, again it is the list returned from read\_in\_file function and returns a dictionary where the key the offense (Arson, Burglary, etc) and the value is another dictionary. This sub dictionary has a key for the zip code, and a value that is how many times this offense occurs in this zip code. In the example below, the vehicular offense occurred in zip code 64161 once and in 64125 once. Arson occurred in zip code 64161 once. Offense is column index 7, and Zip code is column index 13.

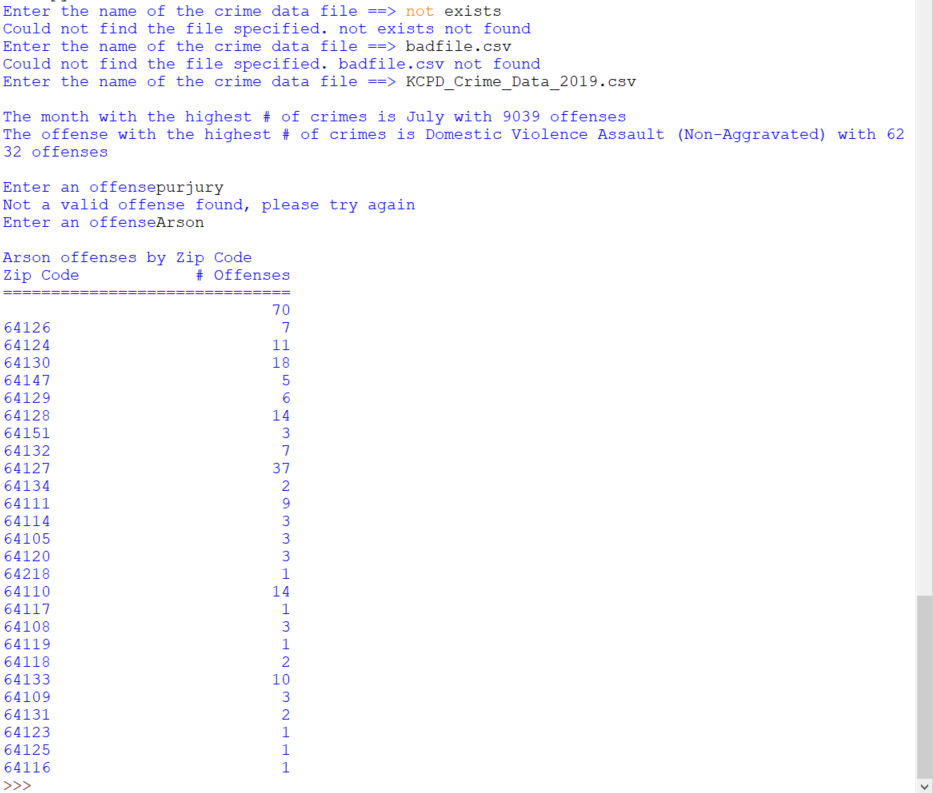


## The final program

Your program should only be in the if \_\_name\_\_ == “\_\_main\_\_” section. Otherwise the unit tests will fail.

The program should ask the user for the file, use a try except so if the read\_in\_file function throws an error you can loop and ask for a filename again. It should then output the month that has the highest crime rate. The offense that occurs the most, and then ask for an offense and output a formatted report of the zip code and how many times that offense occurs in that zip code. It should use the dictionaries created from the functions already given

### Example Output



Notice the error responses to a bad filename or to an offense that doesn’t exist.

## CrimeDataTests.py Unit tests

Run this unit test file to see how well your functions work on other data.

**Grading and Turning In**

Turn in your program before the end of the lab. Only upload the CrimeDataSolution.py file, as other files will be ignored.