Week 3 Assignment Handout

In this weeks assignment, we will be looking at the file 'United States Cancer Statistics, 1999-2011 Incidence.txt'. We will load it into a DataFrame and return some values from it. We will also create a new DataFrame from the data, and plot it. All of this will be done in functions, much like the last weeks assignment was all functions.

Please put this function in a .py file where the file name is 'last_first_week3.py', where 'last' and 'first' are your last and first names.

Please ask questions in the forum if you are confused. There is a wide range of skill levels in this class and I am sure someone knows the answer to your question! Also, **Stack Overflow** is your friend! (Everything you need to know is in this handout and in the lectures, but using stack overflow is very "real world".

1. Create the function loadAndPrepData(filepath). The 'filepath' argument will be a string that gives the path to the 'United States Cancer Statistics, 1999-2011 Incidence.txt' file. This functions does the following:
    a. Reads in the file to a DataFrame using the same read_csv() method function we used in lecture. You will want to use the first row as the header, but none of the rows as the index_col. You will also need to specify the 'sep' argument when calling read_csv(). 'sep' gives the delimiter in the file. The delimiter in this file is not a comma, so it needs to be specified.
    b. Trims off some of the last rows in the DataFrame. This is because these are just notes in the file. Open the file in eclipse and scroll to the end of the file to see what I mean - this will also let you know how many rows you need to remove from the DataFrame. You can do this by creating a new DataFrame from a subselection of the rows from the original DataFrame.
    c. Returns this new DataFrame with the comment rows removed.
2. Create a function called onlyOneState(filepath, state). filepath is the same argument as the above problem. 'state' is a string giving the name of a state. This function needs to:
    a. Return a DataFrame that only contains the rows where the 'State' columns equals the 'state' argument passed to the function.
    b. Remember that since you have already coded the function loadAndPrepData(), you can call that function from within the onlyOneStat() function.
3. Create function called cancerSums(filepath). The filepath is the path to the file, just as above. This function:
    a. Returns a dictionary where the key is a state name, and the value is the sum of the 'Count' column for that state. All states must be included in this dictionary.

b. Remember that you can use previous functions from previous problem to help - you can use those functions within the 'cancerSums()' function.

4. Create a function called stateCountAsList(filepath, state). The arguments to this function are the same as the previous function. This function needs to:
    a. Return a list of the counts for the 'state' passed to the function. To do this, first use the function you coded in problem 2 to get a DataFrame of just one state, then you can grab a list of the 'Count' values with:
        i. countList = df['Count'].tolist()
        ii. NOTE: you need to account for missing data. Some states do not have data for some years (i.e. Alabama does not have data for 1999). When you find a missing year, please use np.nan as the value. Let me explain:

Explanation for using nan from numpy to fill in missing data, in problem 4:

If there is missing data for a year, please use nan from the numpy package. I know this seems like it is coming out of nowhere, I will add an explanation in the handout, and also below.

To use nan from numpy see the following:

```
import numpy as np
list = [1,2,np.nan,3].
print np.nan
```

See that you can use 'np.nan' anywhere you would use a numerical value.
Let's say you were using the follow code to grab 'fill in' a list of counts for 1999-2011

```
import numpy as np
yearList = range(1999,2012)
countsList = []
dfStateSubset = # this is where you would grab a subset of the data for just one state
for year in yearList:
    # code here to see if the year is included in the dfStateSubset data
    # if the year is not in the subset, you would use countList.append(np.nan)
    # if the year is in the subset, you would append the incidence count value to countList
```

5. Create a function called getStateCountsDF(filepath). This function will:
    a. Create a list of lists of the counts for each state.

i. For example, [[1003,205,107], [5006,708,345]]. This is a list of lists, where the inner lists are counts for state. Note that the number here are made up and won't match the actual counts in the data file.

b. Create a DataFrame from this list of lists. You can do this very simply with the following:
   i. newDF = pd.DataFrame(listOfLists, index = ['label1','label2',..]
   ii. See the following screen shot of code:
   iii.

```
>>> b = [[1,2],[3,4],[5,6]]
>>> pd.DataFrame(b,index = ['California','Ohio','New York'])
Out[42]:
            0  1
California  1  2
Ohio        3  4
New York    5  6
```

c. Return this new DataFrame from the function

6. Create a function called plotCounts(filepath). This function will:
   a. Use the function you coded in problem 5 to create the new DataFrame.
   b. Create a line plot of the counts, without a legend. Remember that you may need to transpose the DataFrame - see the lecture for more on this.