Assignment 1

For this assignment, we will be using the baby names database from the Social Security Administration. Use the national database for your responses to the following questions.

1. Compute the total number of births for each year and provide a formatted printout of that.

```
Year Births
1880 23456
1881 12345
1882 13579
...
```

2. Compute the total births each year (from 1990 to 2014, inclusive of both) for males and females and provide a plot for that.

3. Print the top 5 names for each year starting in 1950.

```
Year   Name 1       Name 2       Name 3       Name 4       Name 5
1950   Larry        Sally        Josie        Tom          Dick
1951   Harry        Moe          Mary         Curly        Liz
...
```

4. Find the top 3 female and top 3 male names for years 2010 and up and and plot the frequency by gender.
5. Plot the trend of the names 'John', 'Harry', 'Mary' and 'Marilyn' over all of the years of the data set.
    a. Stack 4 plots one over the other
    b. Plot all four trends in one plot.
6. Find the ten names that have shown the greatest variation over the years. Plot this.

You need to turn in the following functions:

```
getData(folder)

q1(pandasDataset)
      …
      Response to question 1

q2(pandasDataset)
      …
      Response to question 2

…
…

q6(pandasDataset)
      …
      Response to question 6
```

The structure of the program/module that you turn in (one file only) is shown on the next page

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 04 09:09:54 2016

@author: kanungo
GWID: G19860011

A brief description of the program / module not exceeding two lines
"""

import time
import pandas as pd

def getData():
    """Reads multiple files and returns contents in a pandas dataframe.

    Args:
        None:
    Requests for the name of the path for the files in the program
    Returns:
        a list with the file contents
    """

    start_time = time.time()

    # get path name, ending with /
    pathname = input("Please provide the path for the name files ..)"

    # Create empty dataframe; See http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.empty.html
    dfAll=pd.DataFrame({'Name' : [],'Sex' : [],'Count' : [],'Year' : []})

    print ('Started ...')
    for year in range(1880,2016):
        filename = 'yob'+str(year)+'.txt'
        # Read a new file into a dataframe
        df = pd.read_csv(filepath, header=None)
        df.columns = ['Name', 'Sex', 'Count']
        df['Year'] = str(year)
        dfAll = pd.concat([dfAll,df])

    print('Done...')
    print ('It took', round(time.time()-start_time,4), 'seconds to read all the data into a dataframe.)'
    return (dfAll)


def q1(myDF):
    """ Compute total number of births for each year and provide a formatted printout of that

    Args:
        filename: the pandas dataframe with all data
    Returns:
        Nothing
    """
    dfCount = myDF['Count'].groupby(myDF['Year']).sum()
    s = '{:>5}'.format('Year')
    s = s + '{:>10}'.format('Births')
    print(s)
    for myIndex, myValue in dfCount.iteritems():
        s = '{:>5}'.format(myIndex)
        s = s + '{:>10}'.format(str(int(myValue)))
        print (s)

def q2(myDF):

def q3(myDF):

def q4(myDF):

def q5(myDF):

def q6(myDF):
```

How to submit your assignment

1. You need to submit **one** Python file to Blackboard to the **Assignment 1** link.
2. Ideally, you will create, edit and test this file in Spyder
3. The program should be commented well enough so that the TA or I should not have to struggle with understanding variable names and codes and what statements or code blocks do.
4. I will import your module and run it on my machine
5. The grading rubric is shown at the end.
6. Name your file as **`A01_Gwid.py`**. So if your GWID is G19860011 then you should name your file as **`A01_G19860011.py`**.
7. Your program header for the program / module should look something like

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 04 09:09:54 2016

@author: kanungo
GWID: G19860011

A brief description of the program / module not exceeding two lines
"""
```

Rubric for Grading the Programming Assignment

	Unsatisfactory	Satisfactory	Good	Excellent
Delivery	· Completed less than 70% of the requirements. · Not delivered on time or not in correct format (Blackboard or git)	· Completed between 70-80% of the requirements. · Delivered on time, and in correct format (Blackboard or git)	· Completed between 80-90% of the requirements. · Delivered on time, and in correct format (Blackboard or git)	· Completed between 90-100% of the requirements. · Delivered on time, and in correct format (Blackboard or git)
Coding Standards	· No name, date, or assignment title included · Poor use of white space (indentation, blank lines). · Disorganized and messy · Poor use of variables (many global variables, ambiguous naming).	· Includes name, date, and assignment title. · White space makes program fairly easy to read. · Organized work. · Good use of variables (few global variables, unambiguous naming).	· Includes name, date, and assignment title. · Good use of white space. · Organized work. · Good use of variables (no global variables, unambiguous naming)	· Includes name, date, and assignment title. · Excellent use of white space. · Creatively organized work. · Excellent use of variables (no global variables, unambiguous naming).
Documentation	· No documentation included.	· Basic documentation has been completed including descriptions of all variables. · Purpose is noted for each function.	· Clearly documented including descriptions of all variables. · Specific purpose is noted for each function and control structure.	· Clearly and effectively documented including descriptions of all variables. · Specific purpose is noted for each function, control structure, input requirements, and output results.
Runtime	· Does not execute due to errors. · User prompts are misleading or non-existent. · No testing has been completed.	· Executes without errors. · User prompts contain little information, poor design. · Some testing has been completed.	· Executes without errors. · User prompts are understandable, minimum use of symbols or spacing in output. · Thorough testing has been completed	· Executes without errors excellent user prompts, good use of symbols, spacing in output. · Thorough and organized testing has been completed and output from test cases is included.
Efficiency	· A difficult and inefficient solution.	· A logical solution that is easy to follow but it is not the most efficient.	· Solution is efficient and easy to follow (i.e. no confusing tricks).	· Solution is efficient, easy to understand, and maintain.