CIS 41B - Lab 2: Data Analysis/Visualization and GUI, using numpy, matplotlib, tkinter

Write a GUI application that lets the user look up the population growth of countries of the world.

**Input data**

There are 3 input files for the application: *population.csv, countries.csv, years.csv*.

[The files are the cleaned and formatted data from the same [World Bank Open Data](https://data.worldbank.org/) website of lab 1]

* *years.csv* contains the years from 1960-2019, inclusive
* *countries.csv* contains the name, country code, and location of the countries of the world
* *population.csv* contains the population of countries of the world from 1960-2019  
  The data for *population.csv*  is a table of rows and columns as shown in the diagram below
  + Each row has the population numbers for one country, and the country corresponds with the list of country names in countries.csv
  + Each column is the population number for one year

*years.csv*

*countries.csv*

1960 1961 1962 1963 **. . .** 2018 2019

country

names

*population.csv*

**User interaction**

The app gives the user the option to:

1. View the population trend of the regions of the world in the past 60 years.
2. View population trends of user selected countries.
3. View the 10 countries with the largest population in the most current year, 2019.

**Implementation**

The application is divided into 2 files.

* population.py takes care of the data analysis and visualization, and it has 5 main tasks:

1. Read data from all 3 input files and store in a numpy array or Python container as appropriate.
2. Calculate all the static data that support the 3 user interactions shown above. Static data are data that can be calculated ahead of time because it doesn't depend on the user input.
3. Plot the region population trend for all the regions
4. Plot the population trend of user selected countries.
5. Plot the population of the 10 countries with largest population.

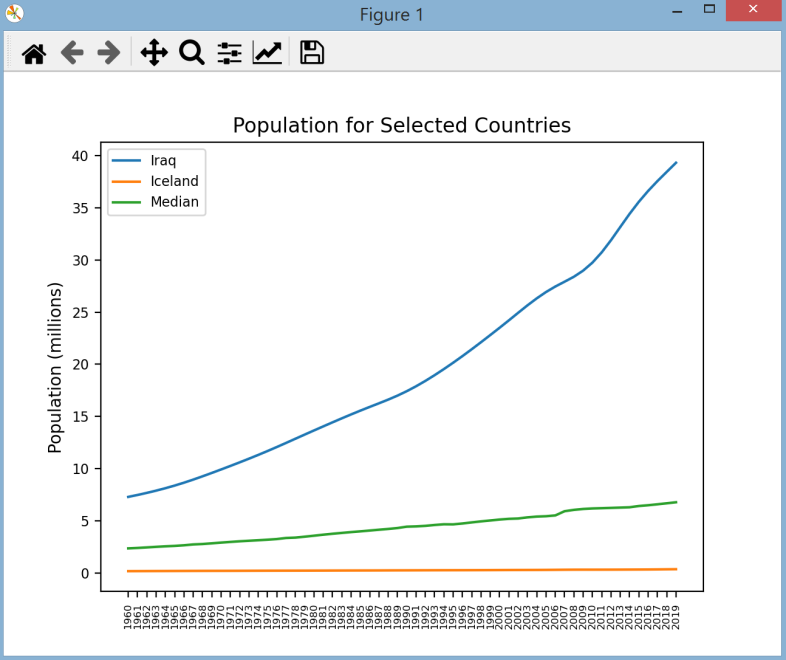
* lab2.py provides the GUI to interact with the user by providing 3 window classes:

1. A main window class that lets the user choose from the 3 user interactions.
2. A dialog window class that lets the user choose one or more countries.
3. A plot window class that shows one of the 3 plots. Note that all 3 plots use the same plot window class.

**Details for population.py**

1. Create a Population class to do each of the following tasks.
2. Read each of the 3 input CSV files into a numpy array or a Python data structure. Note that data should only be read in one time, and then the files are not accessed again.
3. Find the 10 countries with the largest population in 2019, the latest year in the data.
   * Use numpy array features so that your code can be fast and short.
4. Calculate any other static data necessary. If the data comes from a numpy array, make sure to use numpy functions so your code can be fast and short.
5. Plot the population trend for one or more countries.

* The plot method accepts a list of indices, each index is the index of a country in the list of countries.
* From the list of indices, find each corresponding country and plot the population of that country for all the years.
* Also plot the median population of all the countries in the world, for all the years. This helps the user visually compare the population of their chosen countries and the median of all countries.
* The plot must have: a title, a legend that shows the country names, the x-ticks should be the years,   
  a y-axis label, which is the population in millions.
* Example plot:



1. Plot the population trend for each region.
   * For each region, plot the total population of all countries in that region for all the years.
   * The plot must have: a title, a legend that shows the region names, the x-ticks should be the years,   
     and the y-axis label is the population in millions.
   * After plotting, return the names of the region.
2. Plot the population growth of the 10 countries with the highest population in 2019.

* Use the type of plot that makes it easy to compare the population of the 10 countries.
* The plot must have: a title, a y-axis label, and the x-ticks must show the names of the countries.
  + The data should be shown in sorted in order from lowest to highest population.
  + After plotting, return the names of the top 10 countries.

1. Write a decorator called *printResult* that, when applied to a function or method, will print to screen the return value of the function. The decorator should be a global function, not a method of a class.  
   Apply the decorator to the method of steps e and f above.
2. Write a main test driver which will do the unit testing of all the steps above. The main function has 4 lines, each line will call the function/method to:

* Read in data
* Plot the population median and the population for 2 countries of your choice
* Plot the population trend for the regions
* Plot the top 10 largest population and countries

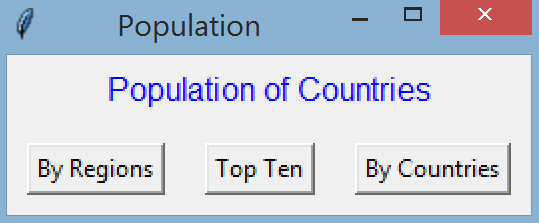
**Details for lab2.py**

1. There are 3 types of window classes for the application, each window *must be a class* which is derived from a Tkinter class.

The sample windows below are just to give you an idea of what the window should have. Feel free to be creative with the look of each window (font, color, size, widget position, text string, etc.)

1. The main window has:

* A title
* A line of text to explain what the application is
* 3 buttons for the 3 types of plot that will be shown (use any appropriate text you like for the buttons)  
  The 3 buttons should be centered with respect to the line of text above them.

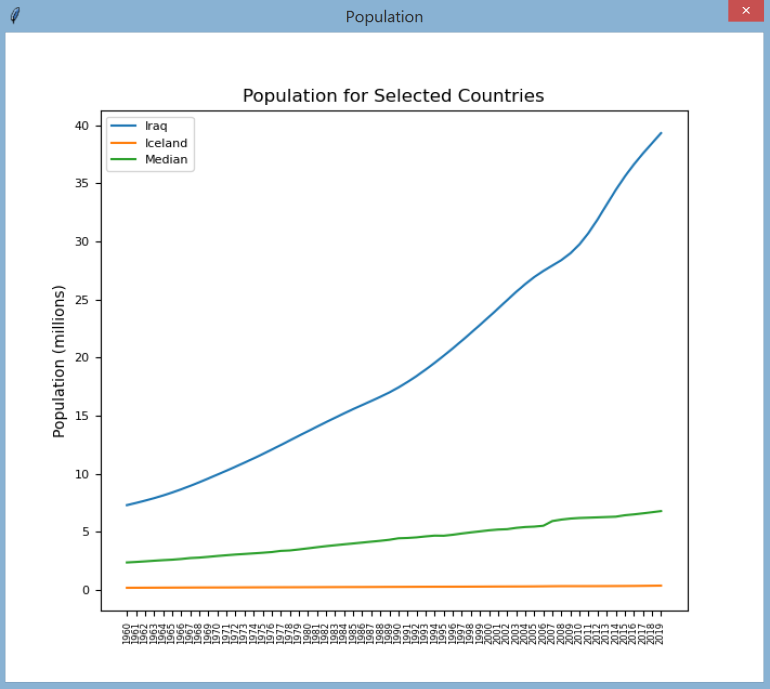


1. The dialog window has:

|  |  |
| --- | --- |
| * A listbox that shows all the country names. * The listbox is sized to show 10 names and has a scroll bar for the user to scroll down the list. In the example window, the scrollbar is about 1/3 of the way down. * There is an OK button centered with the listbox. * The user clicks on each country name to select or unselect it. * When the user is done selecting the country or countries, they click the OK button to lock in their choices. |  |

1. The plot window contains 1 of the 3 plots.

* Make sure you embed the plot inside your plot window object so that the GUI controls all user interface. The plot should not be an independent matplotlib window.
* Note how the plot that was previously shown as a matplotlib window is now in a GUI window in its final form.

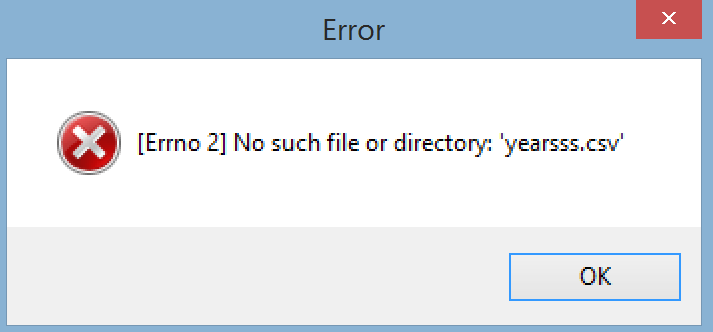


1. Behavior / interaction between the windows:

* At the main window:
  + When the user clicks the "By Regions" button, the plot window appears with the plot for the regions.
  + When the user clicks "Top Ten" button, the plot window appears with the top ten plot.
  + When the user clicks the "By Countries" button, the dialog window appears with the listbox.
  + When the user clicks the X to close the main window, all plot windows that may be opened are closed along with the main window.
* When the plot window appears:
  + The user can go back to the main window to click on a button and open another window.
  + This means there could be multiple plot windows opened at the same time.
  + The user can also click X to close the plot window.
* When the dialog window appears:
  + It should have the focus. The user should not be able to click on any plot window or on the main window to start another event until the dialog window closes.
  + The user can click to select one or more of the countries in the listbox, and then the user must either click X to close the window or click OK to lock in the choice(s) and close the window.
* If the user clicks X to close the window, then no plot window will appear. We assume the user changes their mind and is not interested in the plot. The user is back at the main window.
* If the user clicks OK to lock in the choice(s), then the dialog window closes and the plot window appears with the plot of country population trends. If the user did not select any choice, then no plot window will appear.

1. Exception handling

* During GUI start up, data will be read in from the 3 input files.
* If any of the file open is not successful, a messagebox window will show up to let the user know that there is a file open error, with the specific file name.



* When the user closes the messagebox window, the main window closes.

**Standard additional requirements**

* Have a beginning documentation with your name and a short (one line is okay) description of each file.
* Have a docstring for every public method or function.
* There should be 2 source files, and each source file should work as described above.

**1 pt extra credit for the data analyst in you:**

What are the 2 regions that have the highest population growth in the past 60 years? Why do you think these regions have the largest growth?