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

Write a GUI application that lets the user look up the transfer rate of CA community colleges to the CSUs.

**Input data**

There are 3 input files for the application: *transferCC.csv*, *transferYear*.csv, *transferData.csv*.

[ The files are parts of the downloadable data at: <https://tableau.calstate.edu/views/SelfEnrollmentDashboard/EnrollmentSummary?iframeSizedToWindow=true> ]

* *transferCC.csv* contains all the CA community college names
* *transferYear.csv* contains the academic years for the past 11 years, from 2008-2009 to 2018-2019
* *transferData.csv* contains the transfer rate for each community college for the past 11 years.  
  The data for *transferData.csv*  is a table of rows and columns as shown in the diagram below
  + Each row has the transfer number for one college
  + Each column is the transfer number for one academic year

*transferYear.csv*

*transferCC.csv*

2008-2009 2009-2010 2010-2011 **. . .** 2018-2019

college

names

*transferData.csv*

**User interaction**

The app gives the user the option to:

1. View the trend of the total transfer numbers from community colleges into the CSUs.
2. Compare the transfer numbers of one or more community colleges of the user's choice and the average transfer number.
3. View the schools with the top 10 transfer numbers.

**Implementation**

The application is divided into 2 files.

* transfer.py takes care of the data analysis and visualization, and it has 4 main tasks:

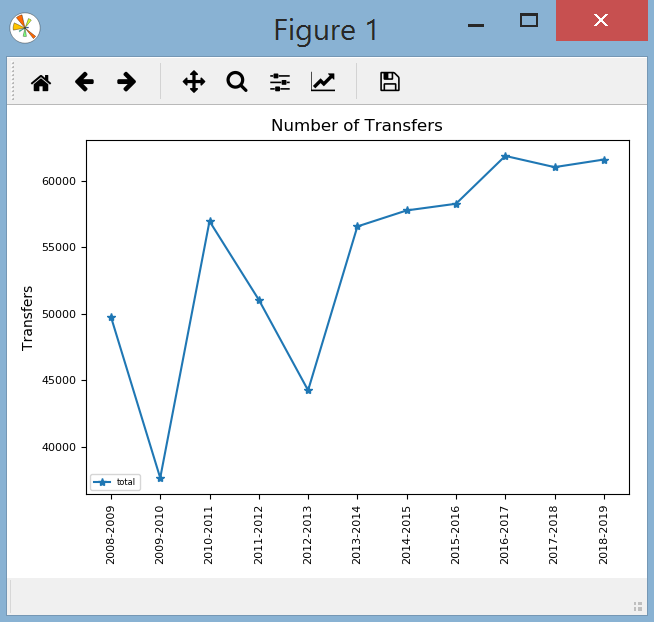
1. Read data from all 3 input files and store in a numpy array or Python container as appropriate.
2. Calculate all the static data that support data User interaction tasks shown above.
3. Plot the total transfer trend
4. Plot the transfer trend of one or more community colleges and of the average of all the colleges.
5. Plot the transfer numbers of the top 10 colleges.

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

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

**Details for transfer.py**

1. Create a 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.
3. Calculate the total transfer rate for each community college by adding all 11 numbers for each college.
   * Use numpy array features so that the calculation can be fast and your code can be short.
4. Calculate the average transfer rate for all the community colleges, one average value for each academic year.
   * Use numpy array features so that the calculation can be fast and your code can be short.
5. Plot the total transfer trend over the given years. The plot must have:
   * A title
   * A y-axis label, and the x-ticks should be academic years



* After plotting the total transfer trend, return the total transfer array.

1. Plot the the enrollment trend for one or more community colleges and for the average enrollment.

* The plot method accepts a list of indices, each index is the index of a community college in the list of community colleges from transferCC.csv
* For each community college in the list of indices, plot the transfer numbers for all 11 academic years. Then plot the average transfer numbers for all 11 academic years.  
  This plot visually compares the transfer numbers of the colleges and the average.
* The plot must have:
  + a title, a legend that shows the college names, a y-label, and the x-ticks should be the academic years
* After plotting, return the average enrollment numbers.

1. Plot the transfer numbers for the 10 colleges with the highest transfer numbers.

* Use the type of plot that easily shows the numbers for individual colleges.
* The plot must have: a title, a y-axis label, and the x-ticks must show the names of the colleges.
  + The data should be shown in sorted in order from lowest to highest transfer number
  + After plotting, return the top 10 transfer numbers.

1. Write a decorator called *printData* that, when applied to a function or method, will print to the output 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 d, e, 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 total transfer trend
* Plot the transfer trend for the average and for 2 community colleges of your choice
* Plot the top 10 transfer numbers

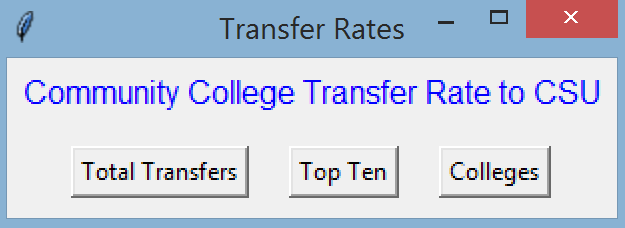
**Details for lab2.py**

1. There are 3 types of windows 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:

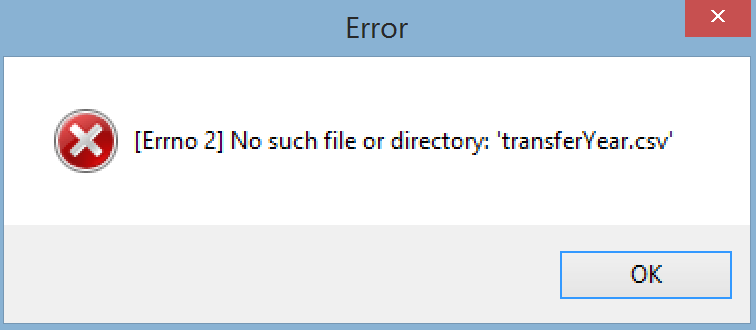
|  |  |
| --- | --- |
| * A listbox that shows all the community college names in alphabetical order. * The listbox is sized to show 10 college names and has a scroll bar for the user to scroll down the list. * There is an OK button centered with the listbox. * The user clicks on each community college name to select or unselect it. * When the user is done selecting the college(s), they click the OK button to lock in their choices. |  |

1. The plot window contains the plot for one of the 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.
2. Behavior / interaction between the windows:

* At the main window:
  + When the user clicks the total transfer button, the plot window appears with the total transfer plot.
  + When the user clicks top ten button, the plot window appears with the top ten plot.
  + When the user clicks the colleges 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 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 college names in the listbox, and then the user must click X to close the window or click OK to lock in the choice 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, then the dialog window closes and the plot window appears with the plot of college transfer trends.

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.



* The user can either click X or click OK to close the error window, which closes the messagebox window, closes the main window, and terminates the application.

**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:**

Notice that there's a substantial dip in transfer numbers in 2009-2010. Search online for a possible reason and explain in your own words what the cause is. Your answer should be 2-3 sentences only.