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

Write a GUI application that lets the user view California community colleges (CCC) enrollment numbers as data plots.

**Input data**

There are 2 input files: students1.csv (smaller file) and students2.csv (larger file). Both files are in the same format.

Each file is a CSV file with columns of data, each column is the number of students at the CCC for one quarter of a particular year. For example, students1.csv has 4 columns of data for Fall 2014, Fall 2015, Fall 2016, Fall 2017.

The first row is the years.

Subsequent rows can be divided into groups of 8 rows:

1. 1st 8 rows: Number of first time students
2. next 8 rows: Number of first time transfer students
3. next 8 rows: Number of returning students
4. next 8 rows: Number of continuing students
5. next 8 rows: Number of unreported students
6. last 8 rows: Number of special admit students

Within one group of 8 rows, each row is for one age groups, in this order:

1. 19 or less
2. 20 - 24
3. 25 - 29
4. 30 - 34
5. 35 - 39
6. 40 - 49
7. 50+
8. Unknown

To view data with respect to their categories, see the file student\_categories.xlsx. The corresponding input file, with numeric data only, is students1.csv. This is the file you should start with.

The file students2.csv is in the same format but with more columns for more years. This file is for extra credit.

[Data source: <https://datamart.cccco.edu/Students/Enrollment_Status.aspx>]

**User interaction**

Give the user the option to:

* View enrollment trend by showing the total number of students for each of the years in a time series plot.
* For one chosen year (out of the available years), view the total number of students for each age group. It's up to you to decide which plot can best show this data.

**Implementation**

The application is divided into 2 files.

* enrolldata.py has 3 main tasks:

1. Read data from a given filename.

2. Plot the enrollment trend as the total number of students vs. years

3. For one given year, plot the number of students in each age group

* lab2.py provides: - a main window for the user to choose the trend plot or the age group plot

- a dialog window to let the user choose a year for the age group plot

- a plot window for each plot

**Details for enrolldata.py**

1. It is recommended, but not required, that you code a class that handles all enrollment data and plots. The class has the following public methods.
2. Read data from the file into a numpy array.
   * Separate the first row into an array of years, the rest of the rows are data for calculation.
3. Calculate and plot the enrollment trend:
   * Add up all data in each column so you get the total number of students for each year
   * Plot the total number of students vs. the years
     + The plot must have a title that shows the year range min-max
     + There should be x-axis and y-axis labels.
     + The x-ticks should be the actual years, the y-ticks be the enrollment in millions (ie. 1.55 is a y-tick instead of 1550000)
     + Use your choice of color and marker, but make sure to show the marker as well as the line connecting the markers (See sample output)
   * Return the array of enrollment numbers
4. Calculate and plot the age groups:
   * Accept a year as input argument
   * Choose the correct year in the array: For each age group: add up all the numbers across the 6 categories of students. You should end up with 8 numbers for the 8 age groups.
   * Plot the number of students per age group, but don't plot the "Unknown"
     + The plot must have a title that shows the input year
     + There should be x-axis and y-axis labels.
     + The x-ticks should show the age range as listed in the first page, the y-ticks are the actual number of students.
     + It's up to you to decide which kind of plot to use.
   * Return the array of age group numbers
5. Add any other methods you see fit.
6. Outside of the class that handles enrollment data, write a decorator called *showNums* that, when applied to the methods of steps 3 and 4, will print to the output screen the array that's returned.
7. Apply the *showNums* decorator to the methods of steps 3 and 4.
8. Write a main test driver which will:
   * Create the enrollment data object with a given filename
   * Call the method of step 3 to see the enrollment trend plot
   * Call the method of step 4 with a chosen year to see the age group plot
   * Call the method of step 4 again with a different year to see the age group plot

Demo step 8 running (further instructions in class) by Mon 10/15, 4:20pm, for 2 pts extra credit.

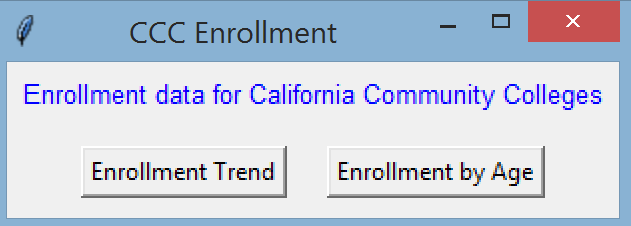
**Details for lab2.py**

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.)

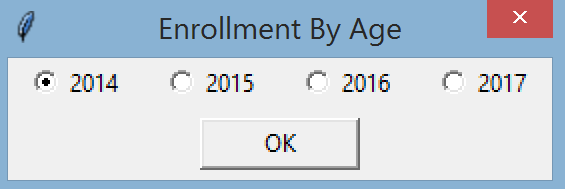
1. The main window has:

* a title
* a line of text to explain what the application is
* 2 buttons: Enrollment Trend and Enrollment by Age (use any appropriate text you like for the buttons)

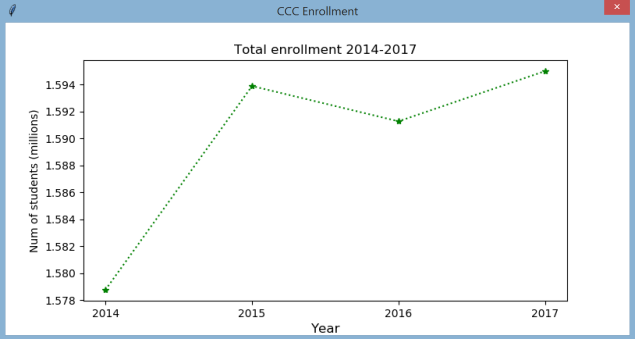


1. The dialog window has:

* radio buttons for each of the years in the data file
* a button for the user to click when the year is chosen and ready to plot (use any appropriate text for the button)



1. The plot window has the plot, for example, the enrollment trend plot.

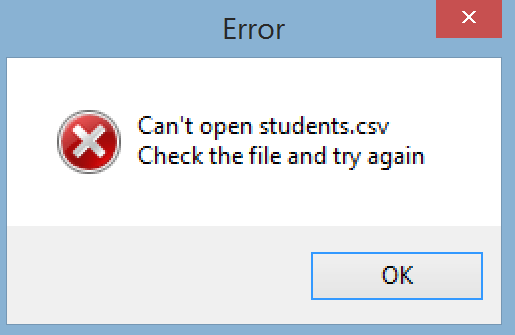


1. Behavior of the windows:

* When the user clicks Enrollment Trend, the enrollment trend plot appears.
* When the user clicks Enrollment by Age, the dialog window appears.
* While the dialog window appears, the main window is disabled. This means the user cannot click on the buttons of the main window and therefore cannot close the main window.
* The dialog window appears with one choice already chosen (2014 in the example above). The user can click to choose a different year. The user must click the OK button for the age group plot to show up.
* If the user does not select OK to plot, and instead clicks X to close the dialog window, then there is no plot window.
* The plot window does not disable the main window. This means the user can click on the main window to open one plot, keep that plot window open, and then click on the main window to open a different plot. The user can choose to have as many plots open as desired.
* When the main window is closed by the user, all plot windows that are opened will close also.

1. Exception handling

* The input file name is a global constant in the lab2.py file. Put the constant at the top of the file so it's easy to change the filename.
* During GUI start up, data will be read in from the pre-defined file.
* If the file open is not successful, a messagebox window will show up to let the user know that there is a file open error.



* The user has 3 choices:
  1. Click the OK in the message box to acknowledge
  2. Click the main window X to close the main window
  3. Click the message box X to close the message box

In all 3 cases, both the main and message box windows close and the GUI ends.

**Lab requirement**

* Have a beginning documentation block with your name, description of each file
* Have a docstring for every public method
* There should be 2 source files, and each of the classes in each source file should be as described above.

**3 pts extra credit for the data scientist in you:**

1. Make sure your code works with both students1.csv and students2.csv without having to change anything, other than the filename constant at the top of lab2.py
2. Run your application with students2.csv so that there are 3 plots on screen:

the enrollment trend plot, the age group plot for 2008, the age group plot for 2018

1. Observe the correlation between the 3 plots.
2. Look up online what happened to the CA economy in 2009. (Suggestion: google "CA economy in 2009" and look at the title of the headlines that show up. You don't have to do deep research, this is not an Economics class)
3. Write a short paragraph (2-3 sentences) in a comment block at the end of your lab2.py file to explain what might be the reason for the 3 data plots on screen.  
   If you enjoy this exercise, you might think about a career in Data Analytics.