CIS 41B - Lab 2: numpy, matplotlib, tkinter

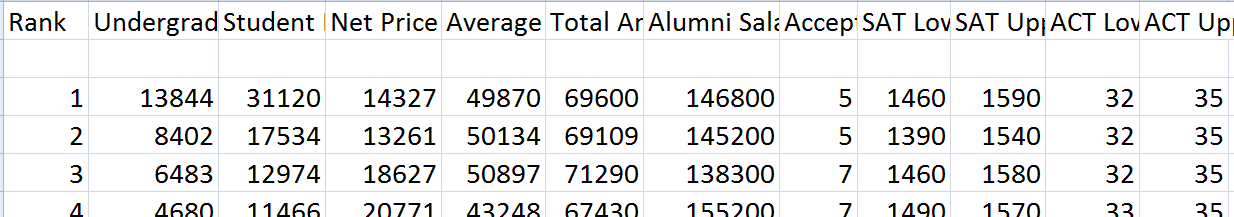
Using the same college ranking data, write a data analysis and visualization GUI application.

**Input files**

The same input data from [Forbes](https://www.forbes.com/top-colleges/#3c36fbc41987) are divided into 3 files:

* colleges.csv: contains the ranking, college name, school type, URL of each college
* scores.csv: contains all the numeric data for each college: student numbers, costs, acceptance rate, test scores
* header.csv: contains the text description for each field (or column) in scores.csv

header.csv



scores.csv

Any missing data field has been filled in with a -1 value.

**Overview**

Lab 1 consists of 2 files that you will turn in: colleges.py, and lab2.py

* colleges.py has a Colleges class, which has data from the input files and methods to analyze and plot the data.
* lab2.py is the GUI that interacts with the user and calls the methods of colleges.py to process the user choice

The user has 4 ways to see college data:

1. A print out of the mean and standard deviation of the cost, lower SAT scores, and lower ACT scores
2. A plot of the distribution of costs of all the colleges
3. A plot of a specific data trend across college rankings
4. A plot the top N alumni salaries, where N is a number

**colleges.py**

Create a Colleges class that will do the following:

1. - Read in data from all 3 files and store them in appropriate data structures of your choice. Choose data structures that can shorten your code considerably, especially if there are numbers involved.  
   - The data files should be opened and read in one time only.
2. - To help the user see the basic requirements to attend one of the top colleges, write a method to find the mean and standard deviation of the Total Annual Cost, SAT Lower, and ACT Lower fields. These 6 values will tell the user the cost and minimum test scores that are needed.  
   - When calculating the mean and standard deviation, don't use the -1 values, which are missing data. For example, if the data values are 10, 20, -1, 30, then only 10, 20, 30 should be used for the calculation.
3. - Write a method to plot the distribution of annual costs so that the user can see the range of annual costs and how many schools are in a certain range.   
   - The plot should have a title, x-axis label, and y-axis label. Make sure the labels are descriptive. 'x' and 'y' and 'numbers' are not descriptive labels.  
   - The method returns the highest and lowest costs.
4. - Write a method that accepts an input argument which is a column number in the scores.csv file. This column number is for one data field (such as Student Population). The method plots the data field against the ranking of the colleges.

- Choose a plot that will show the trend of the data field, if any, as the rank goes from highest rank to lowest rank.   
- The plot should have a title, x-axis label, and y-axis label. The labels should be descriptive, and the title should clearly indicate which field is being plotted. The field name should come from header.csv.  
- The method returns the largest and smallest values of the data field that's being plotted.

1. - Write a method that accepts an input argument which is a number N. The method finds the top N alumni salaries and plot the salaries along with the corresponding school names.   
   - Choose an appropriate plot that shows the school names and salaries. The salaries are plotted in *ascending order*.  
   - The plot should have a title indicating the number of schools, x-axis label, and y-axis label.   
   - Since the school names can be long, make sure the names don't overlap each other on the plot, and they should all fit in the plot.   
   - Since the plot has the highest salaries, it might be helpful to start the plot at $100,000 instead of $0.  
   - The method returns the largest and smallest salaries of the N salaries that are plotted.  
   - See the plot window in the lab2.py section below.

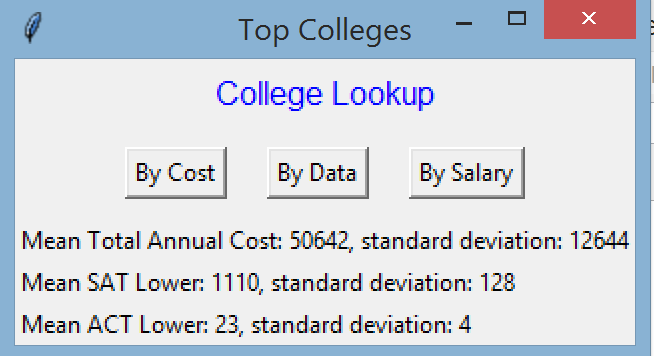
In addition to the Colleges class, write a decorator that prints the return value of the function that it decorates.

Apply the decorator to the 3 plotting methods in steps c, d, and e.

**lab2.py**

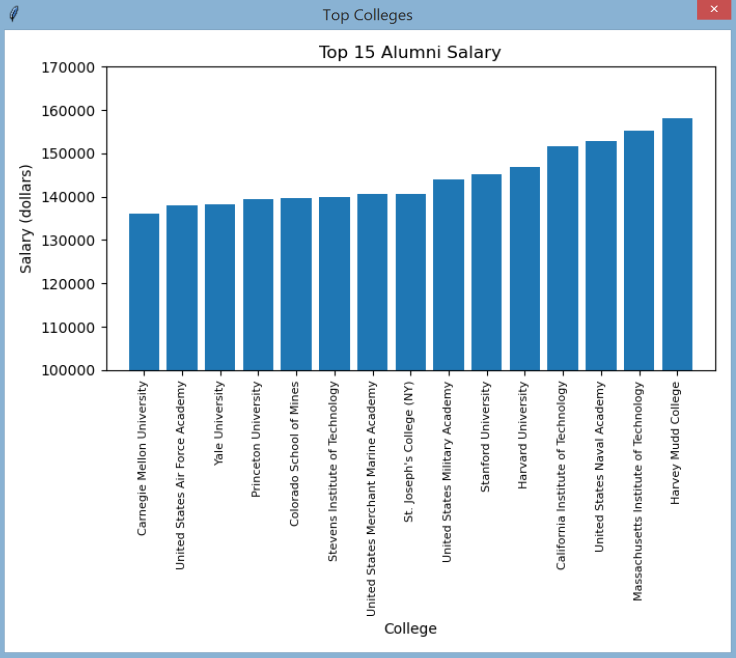
This file contains 3 classes: a main window class, a dialog window class, and a plot window class. Each window class is derived from a tkinter class.

1. The main window is an object of the main window class, and it appears when the app first comes up.
   * The main window class has a class attribute called TOPNUM, which specifies the number of top salaries that will be plotted. Initialize TOPNUM to 15. (When testing your app I will change TOPNUM to another value.)
   * The window has a title, a line of text to explain the purpose of the application, 3 buttons, and 3 statistics about the colleges.
   * The 3 buttons are for the user to see: the cost distribution of the colleges, the data trend for a particular data field, and the TOPNUM alumni salaries.
   * The buttons are centered within the window.
   * The 3 lines of stats are the mean and standard deviation of the cost, lower SAT score, lower ACT scores.
   * The following is a sample main window. Feel free to change the wording of the text strings, and the font size/type/color as you like:



* When the user clicks on the 'cost' button or 'salary' button: a plot window shows up with the appropriate plot (cost distribution or TOPNUM salaries)
  + When the user clicks on the 'data' button, the dialog window shows up to ask the user for a choice of data field.  
    When the user has made a choice, the main window creates the plot window to show the data trend for that data field.

1. The plot window is an object of the plot window class. The plot window is created by the main window:
   * The plot window must be a GUI window, not an independent matplotlib window.
   * There should be one plot window class that can display all the plots of Colleges class. Do not create different window classes.
   * The user can click X to close the plot window, or the user can leave the window open and go to the main window to select another choice.
   * The following is an example of the salary plot window with TOPNUM = 15  
     Note that this is a GUI window and not a matplotlib window.



1. The dialog window is an object of the dialog window class. The window is created from the main window when the user selects the 'data' button.

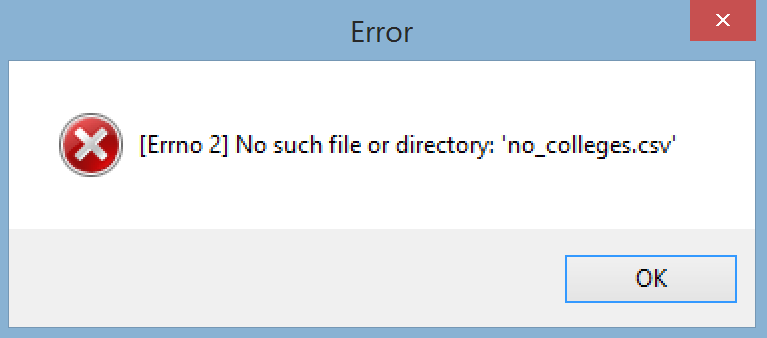
|  |  |
| --- | --- |
| * + The window has a radio button for each data field of the colleges.csv file.   + The text of the radio buttons come from data in the Colleges object, which gets the data from the header.csv file.   + The buttons should be lined up on the left of the window, and the first button is selected by default.   + There is an OK button at the bottom of the window. The user clicks the OK button to confirm their radio button choice.   + When the dialog window opens up, all other windows should be disabled so that the user cannot use the main window to select another choice.   + The user has 2 options with the dialog window: - click OK to commit the radio button choice - click X to close the window and not select a choice |  |

1. Interaction between the main window and the dialog window:
   * If the user clicks the OK button in the dialog window, then the dialog window closes, and the main window can get the user's choice of radio button. Based on the user's choice, the main window creates a plot window to plot the data trend of the user's choice vs. the rankings.  
     Note that the dialog window should not create the plot window. The dialog window's job is to dialog with the user to get the user's choice. The processing of the user's choice is the job of the main window object.
   * If the user clicks X on the dialog window, then dialog window closes and no plot window appears. The user can then go to the main window to make another selection.
2. All the windows are objects of the window classes, so the code should observe OOP standard practices:

* A window cannot directly access private data of another window. Use set/get methods instead.
* Each window should do its own task. For example, the plot window should run its own plot method. The main window should not call the plot method of the plot window.

1. Exception handling

* During GUI start up, data will be read in from the 3 input files. If a file open is not successful, a messagebox window lets 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 and the application terminates.

**Documentation**

- At the top of each file: put your name and a short description of the file (description can be as short as: Colleges class)

- For each public method: add a docstring

You don't need to add docstrings for private method or 'get' method where the method only returns a data attribute, but you're welcome to do so.

.

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

Based on the plots that you run, in a comment block at the end of lab2.py, answer the following questions:

1. Is there a trend for student count vs college ranking?

2. Is there a trend for cost vs college ranking?

3. Is there a trend for alumni salary vs. college ranking?

4. Is there a trend for acceptance rate vs college ranking?

5. Is there a trend for test scores vs college ranking?