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

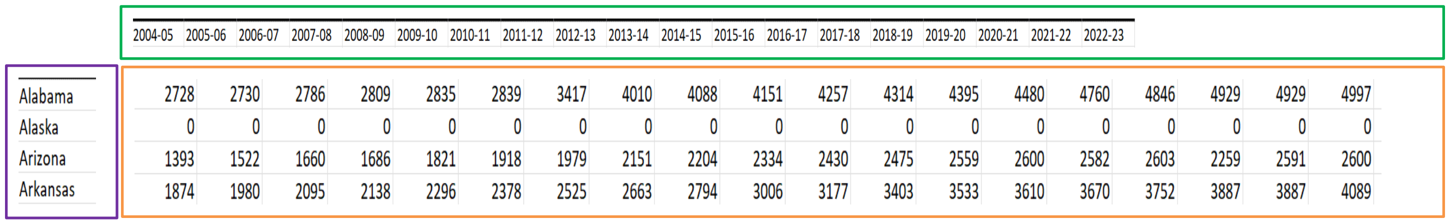
Write a data analysis and visualization GUI application that works with tuition rates at 2-year public colleges in the US.

**Input files**

The input data are from the US College Board (of SAT and AP exam fame) and are divided into 3 files:

- costs.csv: a table of 50 rows and 19 columns of integers
  Each row is for 1 US state (Alaska doesn't have any data so all values are 0 for Alaska)
  Each column is the yearly tuition at 2-year public colleges, and the 19 columns are for the last 19 years.
- states.csv: contains all 50 state names, and each state corresponds with one row of costs.csv.
- years.csv: contains the last 19 academic years, which correspond with the 19 columns of costs.csv.

Here are the first 4 rows of costs.csv, along with the corresponding first 4 states and all 19 years:



| | 2004-05 | 2005-06 | 2006-07 | 2007-08 | 2008-09 | 2009-10 | 2010-11 | 2011-12 | 2012-13 | 2013-14 | 2014-15 | 2015-16 | 2016-17 | 2017-18 | 2018-19 | 2019-20 | 2020-21 | 2021-22 | 2022-23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alabama | 2728 | 2730 | 2786 | 2809 | 2835 | 2839 | 3417 | 4010 | 4088 | 4151 | 4257 | 4314 | 4395 | 4480 | 4760 | 4846 | 4929 | 4929 | 4997 |
| Alaska | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Arizona | 1393 | 1522 | 1660 | 1686 | 1821 | 1918 | 1979 | 2151 | 2204 | 2334 | 2430 | 2475 | 2559 | 2600 | 2582 | 2603 | 2259 | 2591 | 2600 |
| Arkansas | 1874 | 1980 | 2095 | 2138 | 2296 | 2378 | 2525 | 2663 | 2794 | 3006 | 3177 | 3403 | 3533 | 3610 | 3670 | 3752 | 3887 | 3887 | 4089 |

**Overview**

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

- tuition.py has a Tuition class, which has data from the 3 input files and methods to analyze and plot the data.
- gui.py has the GUI class that interacts with the user and calls the methods in tuition.py to process the user choice.

The user has 4 ways to see tuition data:

1. A print out of the minimum, maximum, mean, and median of the 49 current tuition rates
2. A plot of the distribution of tuitions in the most current year
3. A plot to compare the lowest N current tuitions, where N is the user choice.
4. A plot of the tuition trend of the 5 states with the largest tuition change and of the state with the smallest tuition change.

**tuition.py**

Create a Tuition class that has methods to do the following:

a. 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.
   - The data files should be opened and read in one time only.
   - For states.csv and costs.csv: read in all lines into a container (the simplest way to read).
     Then find a way to use all but the 2nd row (no Alaska, since it doesn't have data). You should not have to create a new container without the 2nd row. The new way (excluding the 2nd row) should be used for the rest of the methods.
   - For years.csv, simplify the year by storing only the first year out of each year string. For example, store "2022" from "2022-23".

b. To help the user have an overview of current tuition rate across all the states, plot the tuition distribution.
   - Use the tuition of all states for 2022-23
   - The plot should have a title and axis labels as needed.
   - Return the number of states being plotted. This number should not be hard coded.

c. To help the user see which states will give the best return-on-investment for a 2-year college education, let the user choose the number of states they want to see, and then plot the tuition of the correct number of states that have the lowest tuition rate.
   - Accept a number as input argument. This is the number of states chosen by the user.
   - Choose an appropriate plot so that it's easy for the user to see the difference in tuition among the states. The tuition should be plotted in sorted order.
   - The plot should have a title and axis labels as needed.
   - The state names should be clearly marked with its tuition. The names should not overlap or be too small to view. The names should come from the Tuition object data, don't hard code the names.
   - Return the name of the state with the lowest cost.

d. To help the user see which states have had the largest increase in tuition over the years, plot the tuition trend from the first to last available years for the 5 states with the largest increase in tuition, and then for reference, also plot the tuition trend of the one state with the smallest increase in tuition.
   - Choose a plot that will show the tuition trend of all years between the first and last available years.
   - The 5 state names with largest increase and 1 state name with smallest increase should be clearly labeled. And use a different marker for the 5 states vs 1 state so that the user can easily differentiate the 2 opposite trends.
   - The plot should have a title and axis labels as needed.
   - The state names should be clearly marked with its tuition. The names should not overlap or be too small to view. The names should come from the Tuition object data, don't hard code the names.
   - Return the name of the state with the largest increase.

e. To help the user see some basic statistics on the tuition, find the minimum, maximum, mean, and median of the most current tuition from all the states.
   - Find all 4 statistical values for the 2022-23 year.
   - Return a container with the 4 values, rounded to the nearest whole number.

In addition to the Tuition class, write a <u>decorator</u> that prints the return value of the function that it decorates.
Apply the decorator to the 4 methods in steps b-e, to use for debugging purpose.

<u>Unit testing</u>
It's highly recommended that you write the unit testing code for the Tuition class to confirm that it works, before moving to the ui.py file. The test code should have 5 lines:
1 line to create the Tuition object, and 4 lines to call each of the 4 methods described above.

<u>Lab 2 EC</u>
To really encourage you to do unit testing (a good practice in real life), there will be *2 pts EC* if you show me your code and demo the 5 lines of test code running successfully, at office hour before class (10:30-11:20am) or right after class (1:30pm) on 5/2.

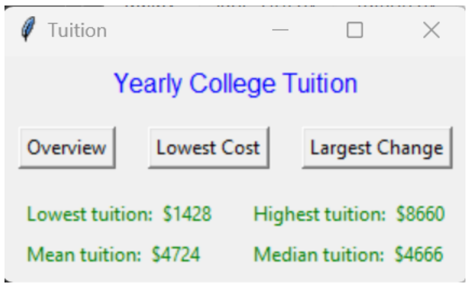
**gui.py**
This file contains 3 classes: a main window class, a dialog window class, and a plot window class. Each of the 3 window classes is derived from an appropriate tkinter class.

The code of the 3 window classes 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 call the plot methods of the Tuition object. The main window should not call these plot methods.
- Each window is responsible for its behavior. For example, when a window needs to close, it should close itself and not rely on another window to close it.

The 3 window classes:

1. The main window is an object of the main window class, and it appears when the app first comes up.
    - The window has a title, a line of text to explain the purpose of the application, 3 buttons, and the 4 statistics about the most current tuition.
    - The 3 buttons are for the user to choose: the overview tuition rates, the N lowest tuitions and states, or the 5 largest changes and 1 smallest change in tuition and their states.
    - The buttons are placed side by side.
    - The 4 statistics are displayed 2 per line, with $ in front of the tuition value.
    - 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:



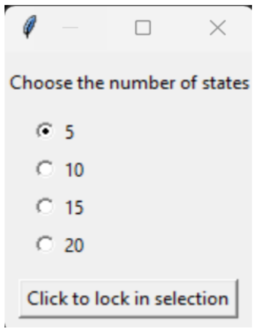
Yearly College Tuition

Lowest tuition: $1428    Highest tuition: $8660
Mean tuition: $4724    Median tuition: $4666

    - When the user clicks on the 'Overview' button or 'Largest Change' button: the main window creates a plot window with the appropriate plot.
    - When the user clicks on the 'Lowest Cost' button, the main window creates a dialog window to ask the user for a choice of number of states to be displayed. When the user selects a choice, the dialog window closes and the main window creates the plot window with the appropriate plot.
    - When the user clicks X to close the main window, all other windows of the app should close.

2. 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 tkinter window that works with matplotlib, and not an independent matplotlib window.
    - There should be one plot window class that can display all 3 plots of the Tuition class. Do not create different plot 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. This means there can be multiple plot windows opened if the user chooses to keep them open.

3. The dialog window is an object of the dialog window class. The window is created by the main window when the user selects the 'Largest Change' button.



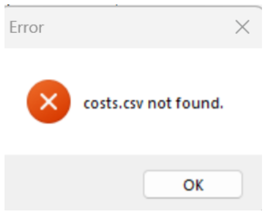
- The window has a radio button for the 4 number choices that the user has.
- The buttons should be lined up on the left of the window, and the first button is selected by default.
- There is a button that the user clicks to lock in their number choice.
- When the dialog window opens, all other windows 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 the button to lock in the radio button choice
  - click X to close the dialog window and not select a choice

Interaction between the main window and the dialog window:

- If the user clicks the button to lock in their selection, then the dialog window closes, and the main window can get the user's number choice of N. Based on the user's choice, the main window creates a plot window with N lowest tuition rates and their states.
  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.

**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.


Error

costs.csv not found.

- 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: Tuition 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.

.