CS 2316 **Individual Homework 5 – NBA Reference**
Out of 100 points

Files to submit: 1. HW5.py

Due before 11:55pm on Wednesday February 17th.

This is an INDIVIDUAL Assignment:

Collaboration at a reasonable level will not result in substantially similar code. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for others.

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use T-Square Forums

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- ***Do not wait until the last minute*** *to do this assignment in case you run into* problems.
- **Read the entire specifications document before starting this assignment.**

Introduction

In this assignment, you will be tasked with reading data from files in CSV format. You will have to find the maximum of various rows and columns of numerical data and display that information on a GUI. Your GUI will also allow the user to select the file from which you should read data.

You will keep the GUI functions of your homework separated from the functions that perform the actual work. The GUI functions will call the "backend" functions that perform the actual work and then display the results. This separation of concerns allows you to change either the GUI or the backend without needing to modify the other.

Your ENTIRE program must be built as a single Python class. Name the class HW5. At the very end of your code file, you will start the TK windowing system and instantiate your object. That may look something like this:

```
rootWin = Tk()
app = HW5( rootWin )
rootWin.mainloop()
```

Break your code up into several manageable methods as outlined below. Please use this suggested program organization and naming scheme, so that we can keep grading consistent.

The breakdown of functionality within your class should be in the following methods:

1. __init__
2. GUISetup
3. fileSelect
4. loadCSV
5. dataManipulator
6. searcher
7. threeLeader

8. freeLeader
9. rbLeader
10. astLeader
11. ptsLeader

File Format Information

For this assignment, you will be given one comma separated value (CSV) file. This file will contain data for every player that has participated in a NBA game this season. In this assignment you will be calculating players' performance on a 36 minute basis. Here's an example of what that means:

Player 1 has played 10 minutes this season and scored 100 points and gotten 50 rebounds. To calculate his 36 minute point average divide his points by his minutes and multiply by 36. The same process is used to calculate his 36 minute rebound average.

Before continuing any further in the assignment please look at the CSV file. The CSV file contains a wealth of information on each player, but we're only interested in a few important statistics. ***In this assignment you will calculate 3 pointers, Free throws, Rebounds, Assists, and Points on a 36 minute basis.***

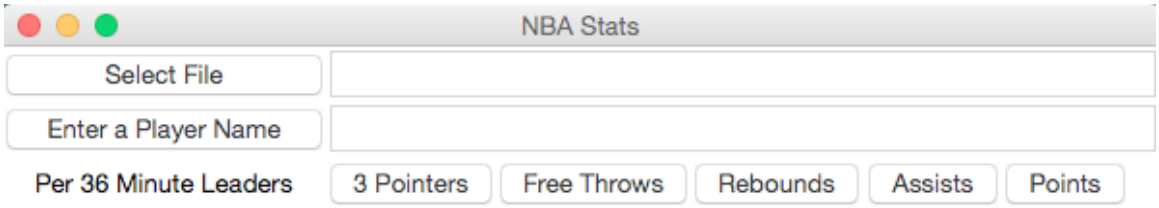
You should parse through the CSV file and store the information in any data type you prefer. You should then calculate their 36 minute average for the above statistical categories, then find the leaders in those categories. In your GUI you will have buttons for the leaders in each category along with a search bar, so the user can find any player's average they wish. You will also have a button that will allow the user to select a CSV file. The functionality of these buttons will be explained below in the description for the methods.

Function Name: __init__(self,rootWin)

This method will be called automatically when your object is instantiated. Often times it is best not to have an overly complicated init function, so in this function you should simply make the rootWin that is assigned to an object variable (hint: use self) and call your GUISetup method.

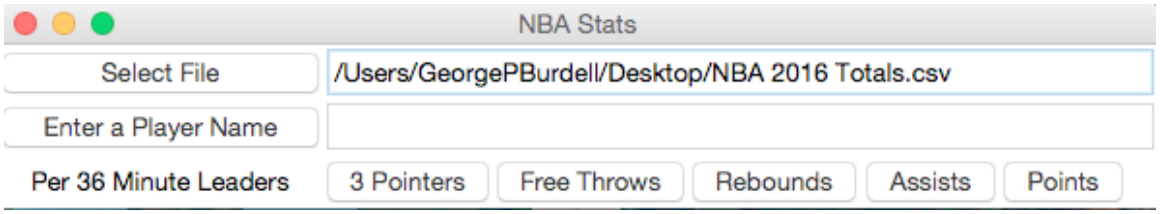
Function Name: GUISetup(self)

In this method you will be creating the GUI below:



Some important features of the GUI:

1. The GUI should have the title: 'NBA Stats'
2. 'Select File' button should call the FileSelect method. It should have a width of 20.
3. The Entry to the right of Select File should be set to 'readonly', have a width of 60 and display the name of the file that has been selected like so:



4. The button with text 'Enter a Player Name' should call the searcher method and the button should be width 20.
5. The entry to the right of the 'Enter a Player Name' button is made for the user to input a players' name. The searcher method description will elaborate on how to treat their submission. The width should also be 60 and the state does not need to be specified because it will default to normal.
6. In the bottom left corner you should have a label with the text 'Per 36 Minute Leaders'
7. To the right of the label you should have a series of buttons. Each button should call their respective leader function specified above. The buttons do not have a specific width.

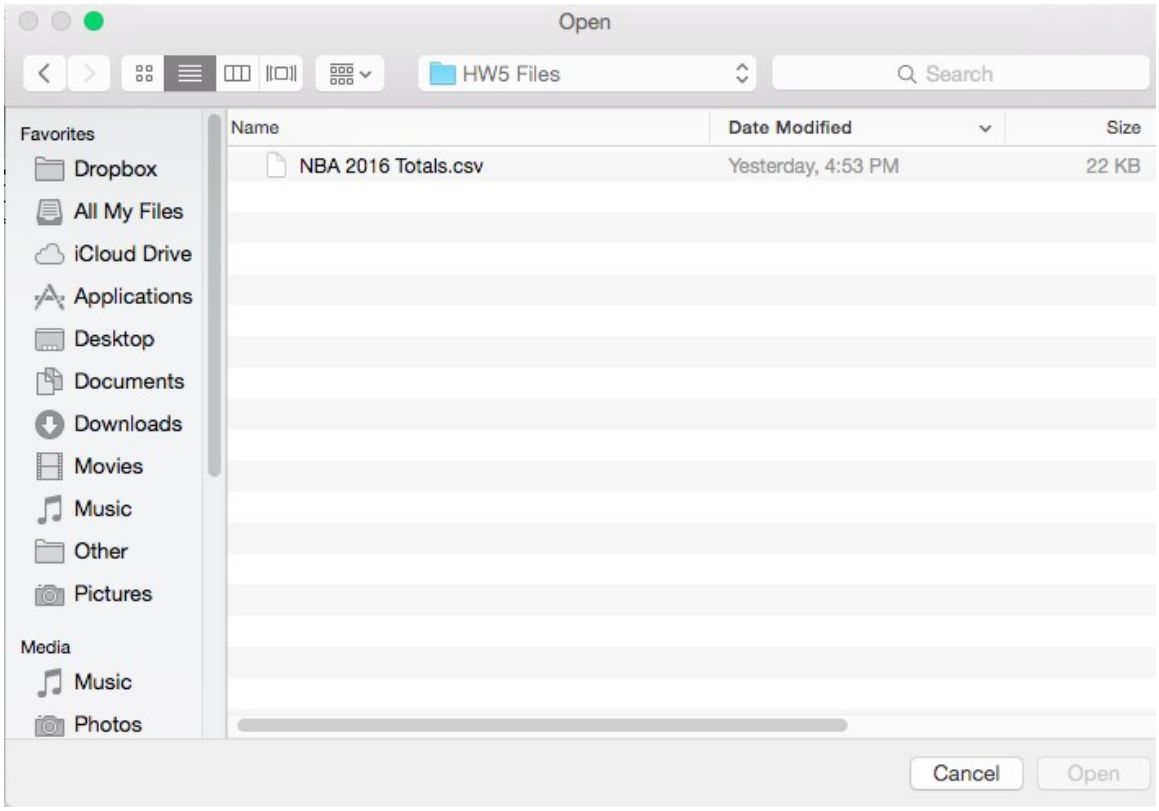
Other useful tips on GUI construction:

- I recommend using grid to arrange the widgets in this gui. The sooner you get comfortable with grid the sooner you'll realize how useful it is.
- If you decide to use grid then one key will be to play around with the columnspan setting. Here's

my best way of describing how to use it:

- In the above gui you have several small buttons that seem to occupy the same vertical area of the GUI as the longer entry. In your mind you can pretty easily see how the buttons and the Label on the left are all divided into one column, let's say that is column 1. Well what if you were to say the big entries were in the same column as the 3 Pointers button, let's say column 2? Then that would mean if you were to put the Free Throws button in column 3 it would be pushed over to the right of the giant entry. If you don't follow me to this point just try to do what I described above. To avoid that from happening you can set the giant Entry to have a columnspan of 5 (it can be any number greater than what you need, often times I will try something like columnspan of 100 then work my way back). If you give the entries a columnspan high enough you should be able to fit all of the buttons underneath it and you GUI will look much better.

Function Name: fileSelect(self)



When the user clicks the "Select File" button, it executes the fileSelect() method. This method will present the user with a file open dialog box. If you do not know how to do this you should read up on tkinter's filedialog specifically about askopenfilename. If the user clicks "cancel" your program should notify the User that they must select a file. If the user actually selects a file and clicks "OK", the fileSelect method will call the loadCSV function. If they do not you should display and error message telling the user to select a file. If you do not know how to do this you should read about messagebox.showerror().

LoadCSV has an output, so it should store what is returned in loadCSV in an object variable. It should then check that loadCSV output a valid list. If it was not able to parse through the file that means there is something wrong and it should notify the user that there is an error in the file they have chosen.

If they picked the correct file, then you should call the dataManipulator method. If they did not select the correct file you should show them an error message. It can be the same message above. If the file is correct it then you should change the entry text to the filename and you're done with this method.

Function Name: loadCSV(self,filename)

Return: csvlist

This function will be pretty similar to ones you have made in previous CSV assignments. I recommend using a try and except here. In the try you should read in the data and in the except you should simply set your output as 'Error File' or whatever value you would like. The purpose of this is so you can properly check for an error in the selectFile method. You should return the list you have created and probably get rid of the first line of the csv that contains the column titles.

Function Name: dataManipulator(self)

This function will do the meat of the calculations. There is no requirement for how you store your data, but this function should do a couple things. It should go through the list created by loadCSV and convert the data from that CSV into the per 36 minute averages we're looking for. If you did not read how to calculate these averages please refer to the file format information section above. ***You should also filter out any player that hasn't logged more than 100 minutes.*** After you have converted the data you should find the leaders of each major stat and save their name however you wish. I recommend storing the players' averages in a dictionary with their name as a key and a list of their averages as the value. You can also have a nested list with elements that contain the player's name and averages. This could be helpful for calculating the leaders, but may be a pain when you're trying to search for a name. You could potentially have both!

If you choose to put the data in a list one way to sort the data would be to utilize python's sort method and pass in a key selection function (you could use a lambda function). If you do not know what this is I encourage you to read about it because it will definitely come in handy.

Function Name: searcher(self)

This method will pull the text the user has entered and try to find the corresponding player's averages. This method should search in a case insensitive manner. Also, to make it a little bit easier on the user you should utilize the in keyword when comparing their entry to whatever list of names you have instead of ==. This will make it so they could enter: stePhen Cur and still get a match (with Stephen Curry). If you are able to match what they have entered you should display their info using a messagebox. The messagebox should look as follows:

If there are multiple matches found, you may use the first one.

Function Names: threeLeader(self), freeLeader(self),rbLeader(self), astLeader(self),ptsLeader(self)

These functions will all basically be the same structure. One could accomplish all of this with just one method, but methods come free with python 3, so why not use them? Each function should output a messagebox that contains the following information:

Grading:

You will earn points as follows for each piece of functionality that works correctly according to the specifications.

General Points

All methods are contained within a class and object is properly instantiated	**10**

The GUI		**40**
Has all labels, text entry fields, and buttons configured as described	10	
Properly allows the user to specify a file to load, and displays the file name when selected.	10	
The data is displayed correctly when the leader buttons are selected	10	
Properly notifies the user when they have errors with their interactions with the GUI	10	

Data Loading / Calculations		**50**
Properly loads data from the CSV file	10	
Correctly converts the string representation of numbers into the per 36 minute averages	20	
Correctly finds the leaders of each category	10	
Correctly matches entries for the player searches as described above	10	