CS 1803

Individual Homework 5 – GUI File Reader

Due: Friday, October 8th, before 6 PM
Out of 100 points

Files to submit: **1. HW5.py**

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 1803, the TA's and the lecturer. Collaboration means talking through problems, assisting with debugging, explaning 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 sum and/or average of various rows and columns of integer data, and based upon whichever column/row has the biggest sum/average, and display that information on a GUI. Your GUI will also allow the user to select the file from which you should read data.

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 have three lines of code that start the TK windowing system and instantiate your object. These lines of code will be similar to the following:

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

Although you are free to organize your class in any way you would like, we *strongly recommend* that you break your code up into several manageable methods as outlined below. If you choose not to follow this suggested program organization, please place a

comment in your code that uses the following names to illustrate where the similar functionality exists in your code:

The recommended breakdown of functionality within your class is the following methods:

1. __init__()
2. clicked()
3. readData(fileName)
4. convertData()
5. findBestDayProducts()
6. findBestAverageProduct()

File Format Information

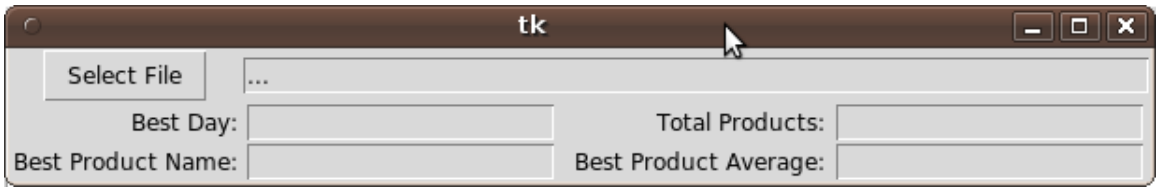
For this assignment, you will be given two comma separated value (CSV) files. These files will have different numbers of rows/columns, but the format will generally be the same. The first row consists of labels for the columns. Each row starts with a day or date, and is then followed by integers representing the number of products produced. See the asotv.csv and dataFile2.csv files for examples.

Your program must work with and produce the correct answers for any similarly formatted data file, regardless of size. You are allowed to use the csv module to read this data, or you may write your own function to read the data.

Here is what we recommend for each of your object methods:

__init__()

This method will be called automatically when your object is instantiated. It is responsible for setting up your GUI. It should create labels and text Entry widgets, along with a button to produce a GUI that looks like the following:

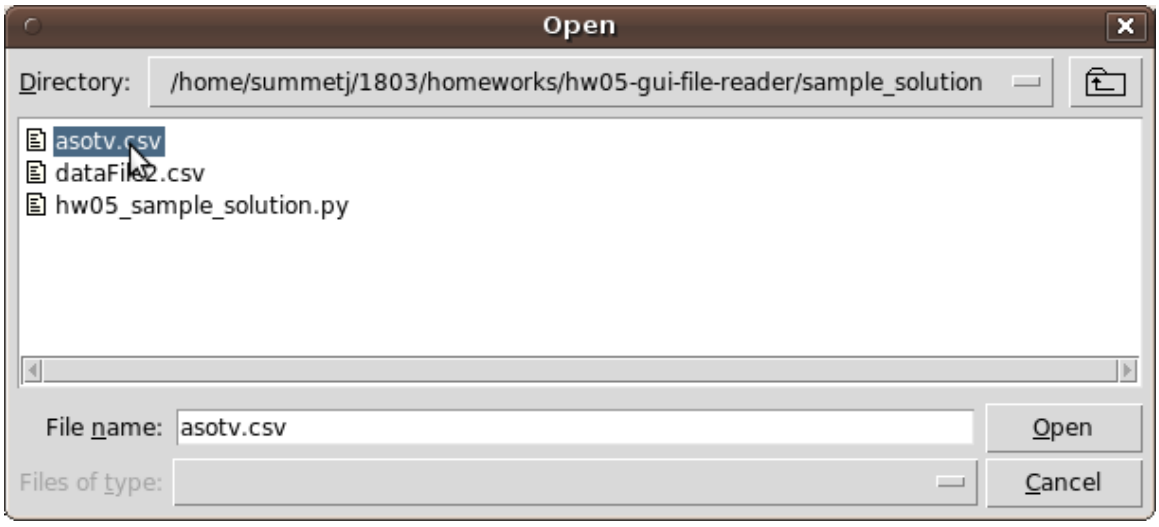


Note that the labels and text entry widgets are nicely arranged. The labels are right justified. The "Select a File" button is directly above the first column of labels. The filename display text entry widget starts out with "..." in it, and spans three columns of data entry display widgets and labels (its width is 60). We suggest you use a grid layout to achieve this effect, although if you are able to do so by using multiple frames and the "pack" layout manager you may use that instead. Note how the labels are justified to the right (or "East") of the grid cells they occupy.

Note that all of the text entry widgets are set to "readonly" at this point.

The "Select File" button should trigger (call) the clicked() method.

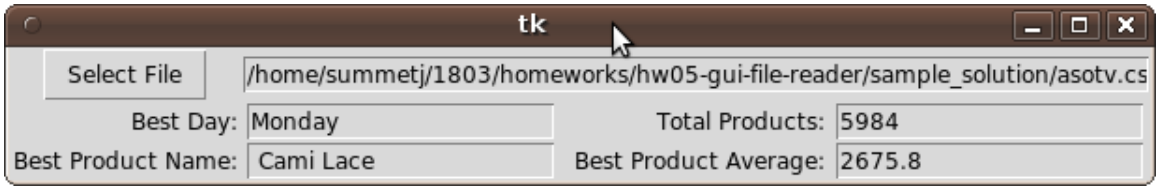
clicked()



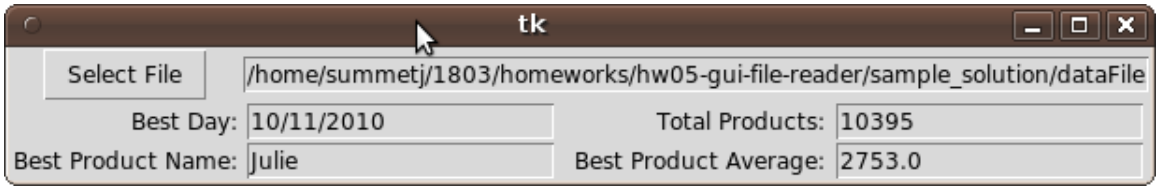
When the user clicks the "Select File" button, it executes the clicked() method. This method will present the user with a file open dialog box. If the user clicks "cancel" your program must gracefully wait for the user to again click the "Select a File" button. If the user actually selects a file and clicks "OK", the clicked method will do a lot more work!

Specifically, it must cause the CSV data to be read in and stored, the textual representation of integers converted to real integers, the sum of rows and the average of columns calculated, and the data from the biggest sum/average displayed to the screen. It will use the readData, convertData, findBestDayProducts and findBestAverageProduct functions to do most of the heavy lifting for it.

Here is an example of the data filled in on the GUI after reading the asotv.csv file:



After loading one file, the user must be able to select the "Select File" button and load in any other file. The GUI must update itself appropriately. Here is an example of the GUI after reading in the dataFile2.csv file:



Note that to change an Entry widget that has been set to "readonly", you must first set the

status to NORMAL, cut the existing text out, insert new text, and then return the status to "readonly". We use Entry widgets so that the user can copy and paste the data out of our program.

readData(fileName)

This function will read in CSV data from the file name passed in. You may assume that the user will only select "valid" CSV files, and do not have to do error checking. You may use the CSV module if you would like. We suggest that you create a member variable (variable attached to the Object, and not local to the function, referenced with the self reference, such as `self.CSVData = []` to store your data. This will allow all methods to access the data via the "self" reference.

convertData()

This method will go through the self.CSVData variable, and convert all of the integers from string representations to actual integers. This will allow you to do math on the numbers in the CSV files. You can either create a different data structure that will be used by the findBestDayProducts and findBestAverageProduct methods, or you can replace the strings with integers in-place. Note that you can not (and should not try to) change the column headers or the dates in the first column to ints.

findBestDayProducts()

This method will find the best "day" by summing up all product production (all numbers in a particular row) for each day, and return the day label (Entry in the first column of the row) and sum of the remaining columns for the row with the largest sum. Note that the very first row of data is a header, so you should only examine the 2nd and subsequent rows.

findBestAverageProduct()

This function will calculate the average product production for each product or worker (depending upon input file). It averages all numbers in a column and finds the column with the largest average. It then returns the top entry in the column (the data label, for the product or worker) and the average for the column.

Grading:

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

The GUI	**40**
Has all labels, text entry fields, and "select file" button	10
Properly allows the user to specify a file to load, and displays the file name when selected.	10
Text Entry areas are "readonly", allowing selection and copying, but not modification by the user	10
The data displayed is updated correctly when each file is loaded	10
Data Loading / Calculations	**60**
Properly loads data from the CSV file	10
Correctly converts the string representation of integers into actual integers that can be used for calculations	10
Correctly calculates the sum of each row	10
Correctly find/displays the label and sum for the row with the greatest sum	10
correctly calculates the average of each column	10
Correctly finds/displays the label/average for the column with the largest average	10