CSSE1001

Semester 2, 2015
Assignment 2
10 marks
Due Thursday 15 October, 2015, 9:30am
Version 1.0.0

A GUI for plotting animal weights and heights

1 Introduction

In assignment 1 you wrote a simple tool for loading and displaying data for arbitrary data sets. For this assignment you will be using animal heights and weights and writing a GUI for plotting this data. The user will be able to open up to 8 data files and plot the data for all the files.

2 Assignment Tasks

For each class and method that you write you need to provide a suitable comment giving a description and where necessary the type and any preconditions. You should use the triple-quote commenting style.

2.1 Download files

The file `assign2.py` must be used for your assignment. When you have completed it you will submit the file `assign2.py` containing your solution to the assignment.

The file already contains some code (in two parts). **Do not modify any of this code! NOTE:** You will lose marks if you don't follow these instructions.

You will also be provided with a zip file, `assign2_files.zip` which contains the following files:

- `assign2_support.py` contains functions and classes that you may find useful when completing your assignment. You should make sure you understand the contents of this file before beginning your assignment.
- Files of the form *AnimalName*.csv, each of which contain sample data that can be used to help develop your application.

2.2 Write the code

Write your solution to the assignment making sure you have included suitable comments. Your solution should include at least the following classes.

NOTE: You must use pack (rather than grid) to do your GUI layout.

2.2.1 AnimalData Class

This class is used to hold the height and weight data for each loaded animal. You are required to use a dictionary to store the data with the animal names as keys and `AnimalDataSet` objects (containing the height and weight data) as values. Because the dictionary does not preserve the order in which the data is entered then you need to keep track of the list of animal names in the order in which they are loaded. Since the user can choose if a given animal's data set is to be displayed, you will also need to keep track of which data sets are currently selected.

This class must define at least the following methods.

- `__init__(self)` that initializes the internal data.
- `load_data(self, filename)` loads in data from the given filename (of the form *AnimalName*`.csv`). Note: This method should not load a data set if a set with a similar name has already been loaded.
- `get_animal_names(self)` returns a list of animal names (as strings) in the order they were loaded in.
- `get_animal(self, animal)` returns a reference to an AnimalDataSet object given an animal's name.
- `is_selected(self, index)` informs as to whether the `index`th animal data is to be displayed.
- `select(self, index)` sets the `index`th data set flag, indicating that the animal data should be displayed.
- `deselect(self, index)` clears the `index`th data set flag, indicating that the animal data should not be displayed.
- `get_ranges(self)` that returns a 4-tuple of the form
  `(min_height, max_height, min_weight, max_weight)`.
  Note: This method should only return the range of data sets that are currently selected. If no sets are selected then a `(None, None, None, None)` should be returned.
- `to_tabbed_string(self, index)` returns a padded string summarising the `index`th data set.

Example:

```
>>> data = AnimalData()
>>> data.load_data("slow loris.csv")
>>> data.get_animal("slow loris")
AnimalDataSet(slow loris)
>>> data.get_animal_names()
['slow loris']
```

```
>>> data.to_tabbed_string(0)
'slow loris     53        Visible   '
>>> data.load_data("bandicoot.csv")
>>> data.get_animal_names()
['slow loris', 'bandicoot']
>>> data.get_ranges()
(11.1, 71.9, 0.514, 2.986)
>>> data.deselect(1)
>>> data.is_selected(1)
False
>>> data.is_selected(0)
True
>>> data.get_ranges()
(11.1, 19.7, 1.303, 2.986)
>>> data.deselect(0)
>>> data.get_ranges()
(None, None, None, None)
>>> data.load_data("slow loris.csv")
>>> data.get_animal_names()
['slow loris', 'bandicoot']
>>>
```

2.2.2 Plotter Class

This class is responsible for doing the plotting and should inherit from `Canvas`.
This class should contain a method capable of plotting data points. Each point is to be plotted as a 5 pixel by 5 pixel square centered at the x and y canvas coordinate corresponding to the point's height and weight. The colour of each point is dependent on the index of the animal that it belongs to and the corresponding colour in the `COLOURS` list provided in the support file.
A vertical and horizontal line should overlay the canvas and intersect at the mouse location when data is being displayed.

2.2.3 SelectionBox Class

This class is responsible for displaying the list of animal data sets and should inherit from `Listbox`.
Each animal data set should be displayed in its own line in the `SelectionBox`. Each line will contain the set name, number of points and selection status. The colour of each line is similar to that of the data points from the same set.

2.2.4 AnimalDataPlotApp Class

This is the top-level class for the GUI. It is responsible for creating and managing instances of the above classes. This class should also govern the layout of the main application window. That is, it should instantiate the necessary `Button` and `Label` widgets and define any necessary callbacks.

At the top of the application there is a file menu to be used for opening station data files. The "Open" menu choice should use `filedialog` so that the user can choose a file to open.

An outline for the GUI layout of the application can be observed in the screenshots provided on the course website.

Note that, if the user enters an invalid data file (one that is not readable, one that does not end in '.csv' or one with invalid data) then an exception will be raised. You need to catch exceptions and display them in a pop-up `messagebox` window.

When a user opens a valid file, the data is displayed in the canvas window and an addition is made to the listbox.
When a data set is selected (or deselected), the points on the graph are to be adjusted accordingly. If no data set is selected when a button is pressed an error window should appear, alerting this to the user.
When the window is resized, the position of points on the canvas should be adjusted accordingly.
The `SelectionBox` widget should not stretch or expand horizontally as the window is resized.
When a user drags their mouse across the canvas widget, the height and weight of the mouse location should be displayed in the label directly above the canvas.
If no data sets are selected, no 'crosshair' should be displayed on the canvas and the highlighted height and weight label should be empty.

2.3 CSSE7030

CSSE7030 students will be required to add a statistic summary window to their application, as well as an additional button in their main GUI window to allow access to this.

Students should add a "Summary" button in line with the "Select" and "Deselect" buttons above the selection box which opens the summary window when there is an animal selected and no window is already open.

2.3.1 SummaryWindow Class

SummaryWindow is the class representing this statistic summary window, and responsible for updating the displayed summary values. It must inherit from `Toplevel`.

The window should display the name, count, and weight and height mean and standard deviation of the currently selected animal as per the example images provided. It is recommended that students use Python's built-in `statistics` package from these. It is important to note that as per assignment 1, we are wanting population standard deviation, not sample.

This summary window should be linked to the state of the selection box, so that whenever the user selects a new dataset in the selection box, the summary window should change to match this, with a maximum of one open summary window at a time.

2.4 Examples

The course web page contains screenshots and a screencast of the application.

2.5 Hints

For redrawing, either because the window has been resized or because the data to be displayed has changed, it's easiest to delete everything on the canvas (`delete(tk.ALL)`) and delete all the data displayed in the data frame and draw all the required lines.

For opening a file from the `File/Open` menu use `filedialog.askopenfilename`.

Week 9 and 10 tutorials discuss GUI layouts and are very useful when attempting to complete this assignment.

3 Assessment and Marking Criteria

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in the practical session you have signed up to in week 12. You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution;
- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. A technically correct solution will not elicit a pass mark unless you can demonstrate that you understand its operation.

We will mark your assignment according to the following criteria.

Criteria	**Mark**
Your code is mostly complete, correct, clear, succinct and well commented. You are able to explain your code.	8 - 10
Your code has some problems OR you have some problems explaining your code.	4 - 7
Your code is clearly incomplete, incorrect, too complex or hard to understand OR you have major problems explaining your code.	1 - 3
Your work has little or no academic merit.	0

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out that code and we will mark that.

Please read the section in the course profile about plagiarism.

4 Assignment Submission

You must submit your completed assignment electronically through Blackboard.

Please read `http://www.library.uq.edu.au/ask-it/blackboard-assessment` for information on submitting through Blackboard.

You should electronically submit your copy of the file `assign2.py` (use this name - all lower case).

You may submit your assignment multiple times before the deadline - only the last submission will be marked. After each submission please use Blackboard to check that the file you submitted was the one you intended to submit. Make sure the file is called `assign2.py` and not, for example, `assign2.py.py`

Late submission of the assignment will not be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on-time, you may submit a request for an extension.

Requests for extensions should be made as soon as possible, and preferably before the assignment due date. All requests for extension must be submitted on the UQ Application for Extension of Progressive Assessment form:
`http://www.uq.edu.au/myadvisor/forms/exams/progressive-assessment-extension.pdf`
no later than 48 hours prior to the submission deadline. The application and supporting documentation (e.g. medical certificate) must be submitted to the ITEE Coursework Studies office (78-425) or by email to `enquiries@itee.uq.edu.au`. If submitted electronically, you must retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.