Unit 10 Exercises – Lists, Functions, Graphics

Aims and objectives

- Working with nested lists, and with graphics
- Systematic testing
- More advanced use of the functions from the Canvas module

This week's exercises

These exercises are based on ideas covered briefly in Chapter 9 of the book, and in recent lectures which you should re-read.
As usual, you will find it useful to do some planning on paper before you start working on the machines.

Task 1 – The *join* function (and *split*)

a) Write a function *join* which, given two lists, it returns a list in which each element is a list of two elements, one from each of the given lists. For example:

```
join( [1,2,3] , ["a","b","c"] )
```

returns

```
[ [1,"a"], [2,"b"], [3,"c"] ]
```

We assume that the given lists both have the same length.

b) Write tests for the function: Each test should print two lists and the result of applying *join* to them. You might like to use the idea of generating test cases randomly, as discussed in previous lectures. Also, make sure that you test the case of two empty lists, the case of two lists with just one element each, and a range of other lengths.

c) Write *split*, the opposite of *join* which, given a nested list where each element is itself a list of two elements, it breaks it down to two different lists.

Problem – drawing a bar chart

Read the problem description, then work through Task 2 and Task 3 below.

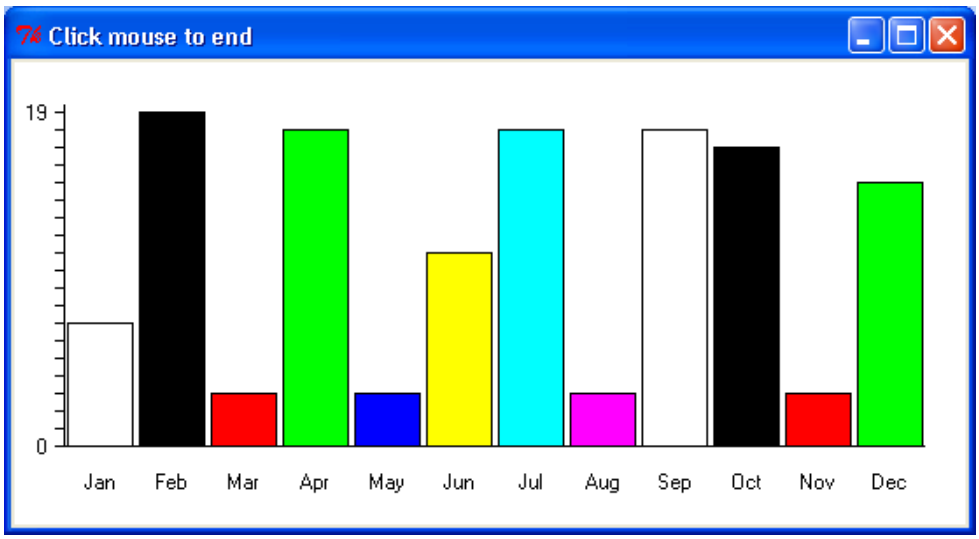
The problem is to write a function *barchart* that will draw a bar chart similar to the one shown below, taking all the relevant information as parameters.

The first line of the function definition should be

```
def barchart(xOrigin,yOrigin,xSize,ySize,labels,data):
```

The meaning of the parameters is as follows.

- *xOrigin, yOrigin* are integers which are the coordinates of the origin of the chart (the point where the axes meet)
- *xSize, ySize* are integers which are approximately the width and height of the whole chart. Actually *xSize* is the length of the x axis and *ySize* is the height of the tallest bar.
- *labels* is a list of strings which are the labels below the bars in the chart.
  In the example: `[ "Jan", "Feb", "Mar", ... ]`
- *data* is a list of integers which are the heights of the bars.
  In the example: `[ 7, 19, 3, 18, ... ]`



Important: the tallest bar must use the full height of the chart, as specified by *ySize*. The bars must use the full width of the chart, as specified by *xSize*.

To help you develop the *barchart* function, it is separated into Task 2 and Task 3. You should do as much planning as possible before you start typing code on the machine.

Task 2

This task focuses on the calculations needed to work out the width and height of each bar. Note the following points:

- The width of each bar should be calculated from *xSize* and the number of data items
- The heights of the bars should be calculated by multiplying the data items by a scaling factor, which depends on *ySize* and the largest data item (write a separate function to calculate the largest item).

Using the function definition (first line) above, write a function that outputs (prints on the screen) the following information, calculated from the parameters of the function. The function does not need to display any graphics at this stage.

- The number of data items.
- The width of each bar, in order to make the total width of the chart equal to *xSize*.
- The largest data item.

- The scaling factor *yScale*, so that the height of each bar can be calculated by multiplying *yScale* by the data item, and so that the tallest bar has height *ySize*.

Task 3

This task extends the *barchart* function from Task 2, replacing the text output with the display of the chart itself. The following points will help you.

- The labels can be displayed using the function *create_text* from the Canvas module: for example, `create_text(100, 100, text="Hello").` The coordinates are the position of the centre of the text. This uses a new method of giving a parameter to a function, which we will learn about later; for now, just use it.
- The bars can be drawn using the function *create_rectangle* from the Canvas module, which you already know about. To specify a colour, use a call such as `create_rectangle(100, 100, 150, 150, fill="red").` The available colours are: black, white, red, blue, green, yellow, cyan, magenta. Think about ways of making each bar a different colour **without** using a separate line of code for each bar (because you don't know in advance how many bars there will be, so of course your function must use a loop related to the list of data items).

Test your function with various lists of labels and data, including lists of different lengths and data with different ranges of values. A good way of testing could be to generate lists of random data, which you can do by using the *random* function from the random module.

If your data includes 0, you might notice something a little odd happening when the corresponding bar is drawn. You might like to try to understand this, although it is not essential to do anything about it for this exercise.