CO1111 / CO1115 Computing Skills  
Module Tutor: Dr Nick Mitchell [NPMitchell@uclan.ac.uk](mailto:NPMitchell@uclan.ac.uk)

**Lab Worksheet 2: Going Further with App Inventor – Variables, Lists and Conditionals**

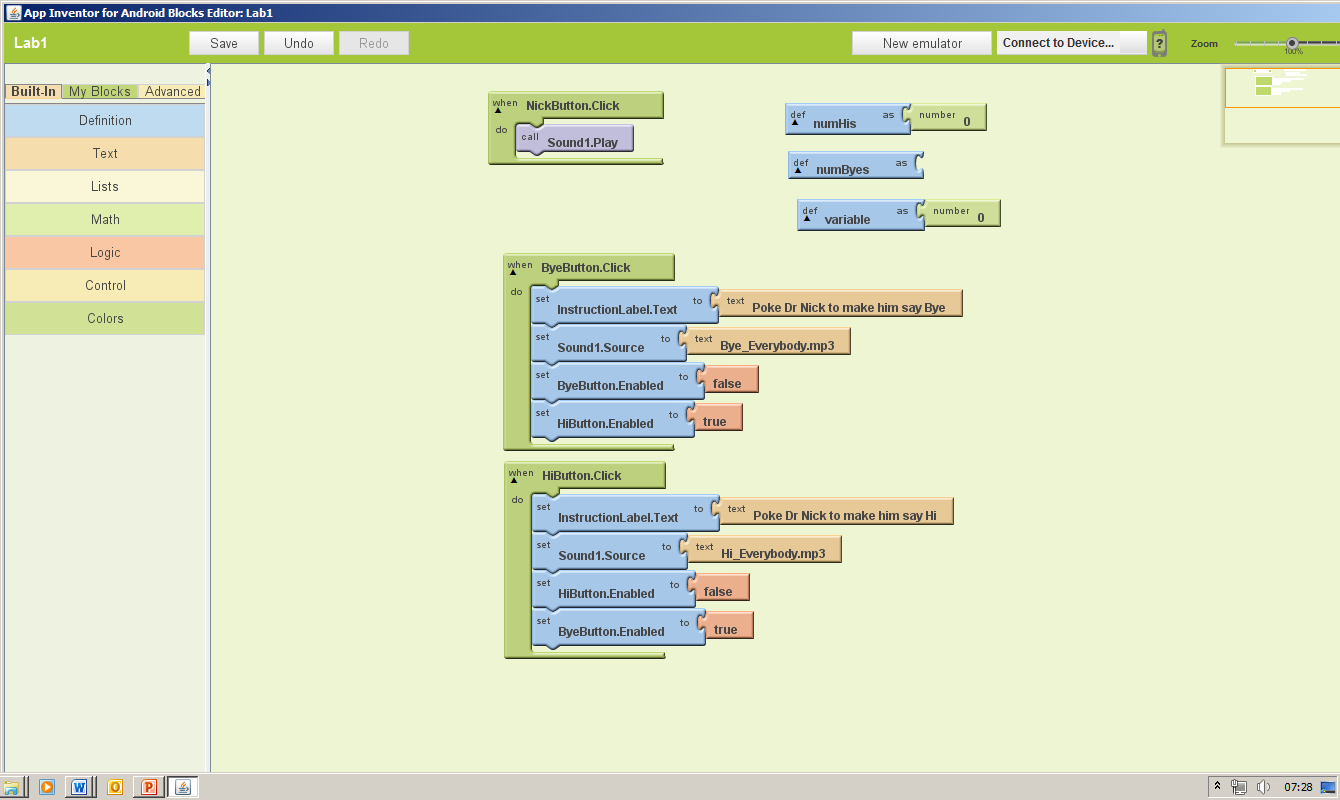
**This worksheet assumes you have completed Lab Worksheet 1, and are conversant with the basic workings of App Inventor.**

This worksheet will take you towards making a multiple choice quiz to satisfy this week’s Challenge.

You may wish to work through the first parts of the exercises individually, to become familiar with the concepts of lists and ifs. Later on, you may wish to work with someone from your team to bounce ideas around about the challenge.

A tutor will be in the lab for the whole session again today in case you get stuck – there is also an optional drop-in session this evening (see your timetable) in case you need extra help after the lab.

Log in at <http://www.appinventorbeta.com> and create a new project. Give it any name you like, but make sure the name is meaningful. Something like Lab2 would be OK if you’re stuck for inspiration.

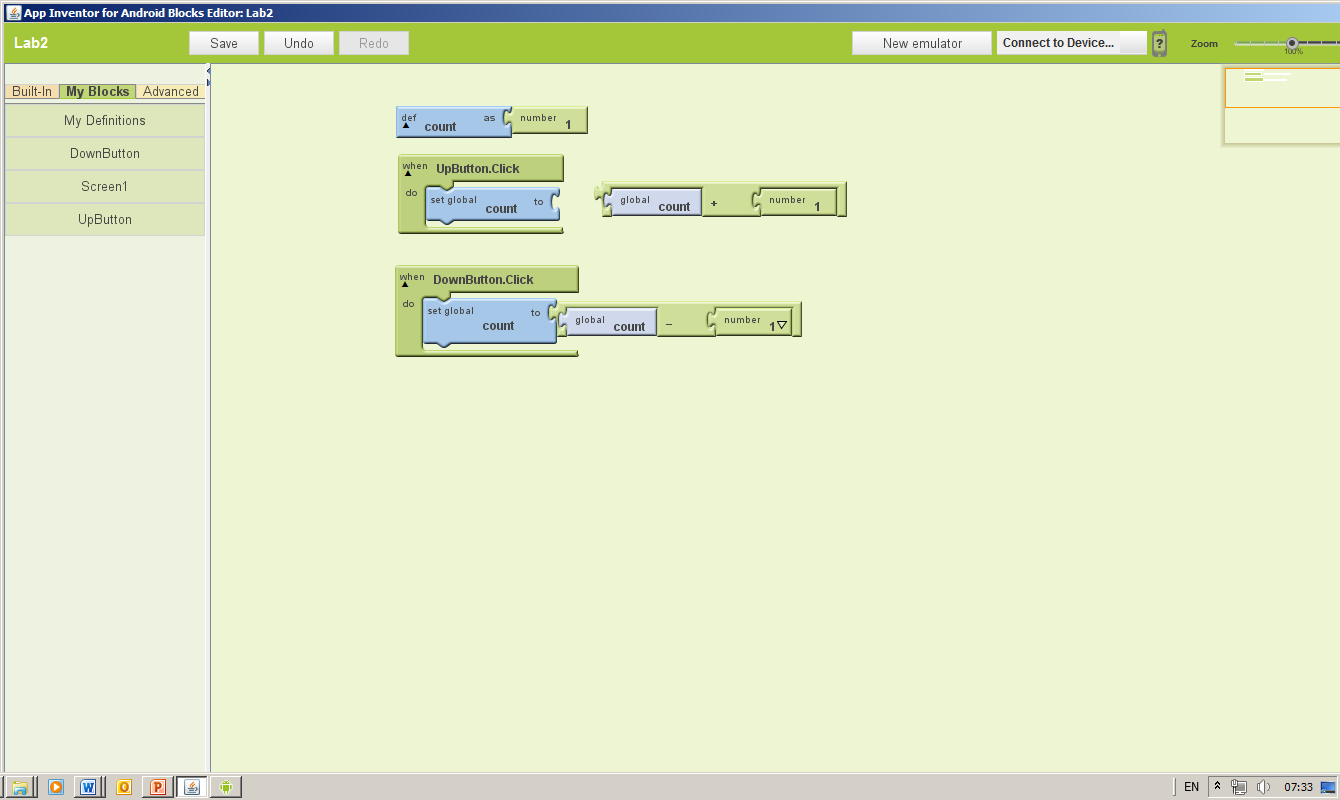
Start by creating an interface with just two buttons. Rename them UpButton and DownButton, and set their text properties to Up and Down respectively.

In the blocks editor, define a global variable called Count (get the block from the Definition drawer), and initialise it with the number value 0.

Drag out event handlers for when each of the buttons is clicked.

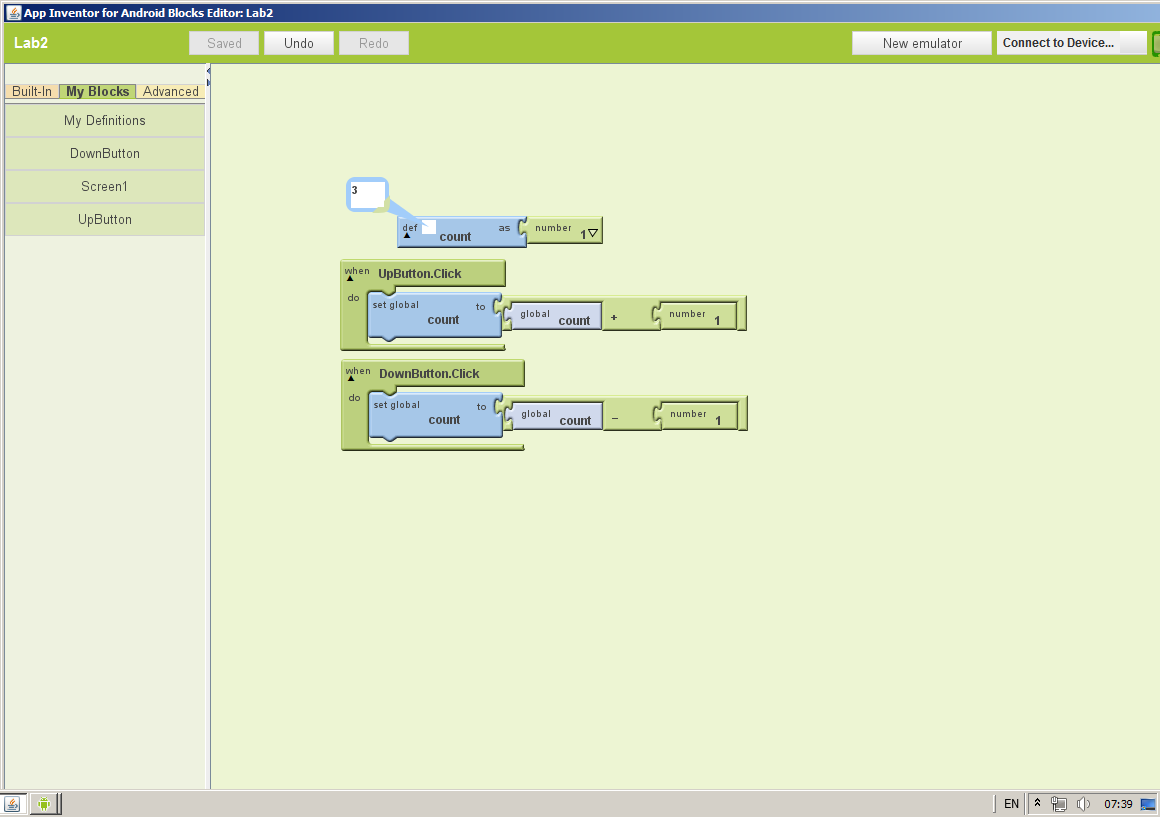
Fill in the event handler blocks so that when the Up button is clicked, the value of the variable is increased by 1. Similarly when the Down button is clicked, the value should be decreased by 1.

To do this, you will need to build an *expression* that will be the new value to assign to the variable.

In the Maths drawer, you will find the jigsaw piece for adding two numbers together. In one slot you need to put the original value of Count, and in the other the number you want to add (i.e. 1)

You can then set value of the variable Count to this expression inside your event handler.

Now run this program in the emulator. Is it working?

From the interface, it is impossible to tell, because the variable is hidden inside the program, and a change in its value has no visible effect. You can, however, “watch” the variable inside the blocks editor to see the value at all times, by right-clicking on the variable and selecting Watch.

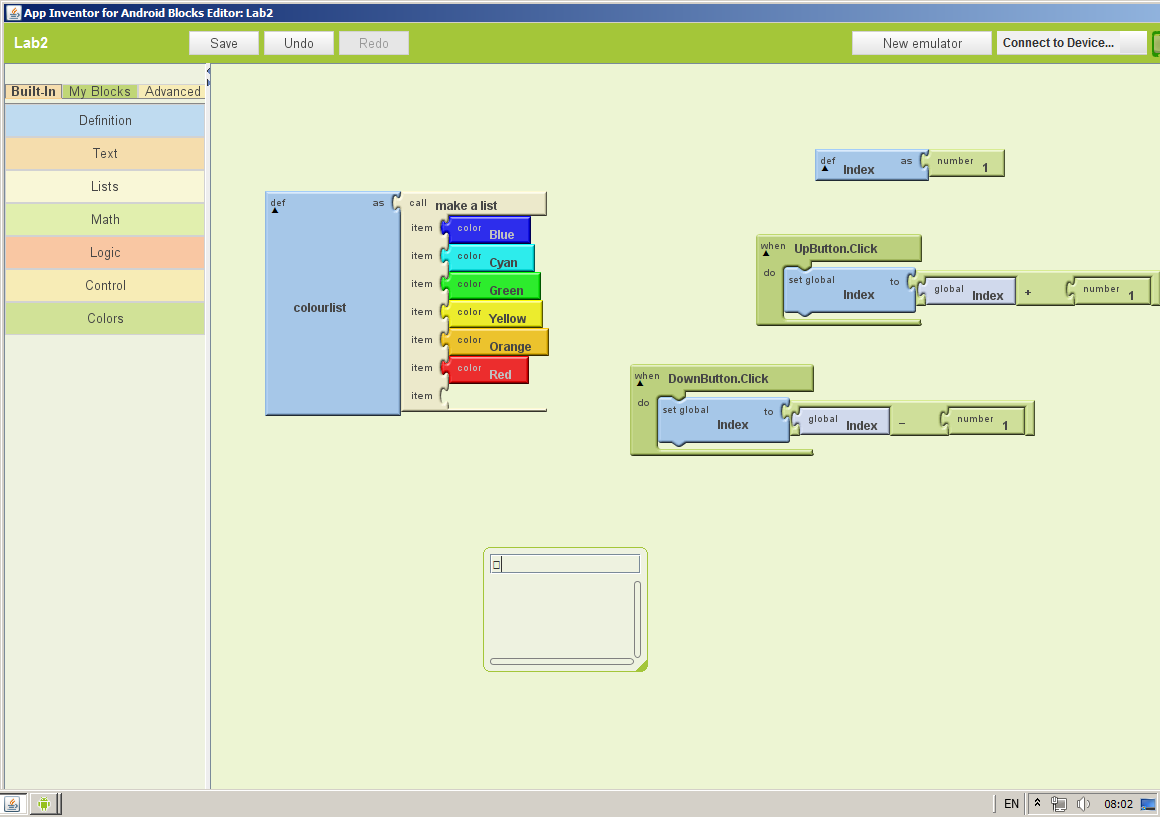
Place a watch on Count, and verify that your program is working correctly.

You can either start a new project, or modify your existing one for the next part.

Once again you need two buttons, and also a label this time. The buttons should be named NextButton and ReverseButton. The label should be called ColourLabel. Set it to be the full width of the screen, and change the background colour to blue.

The app you will build should cycle the background colour of the label through a sequence of colours (blue, cyan, green, yellow, orange, red), in order, moving on to the next colour each time the Next button is clicked.

Once that is working, we will add functionality so that each time the Reverse button is clicked, the direction through which the colours are cycled is reversed.

We will use a list to define the sequence of colours. Drag out a MakeList block from the Lists drawer, and plug the sequence of colours into it.

The list of colours is a value (just like a number is a value, but more complex). It needs to be stured in a variable so it can be accessed.

Define a variable called ColourList, and plug the list of colours into it.

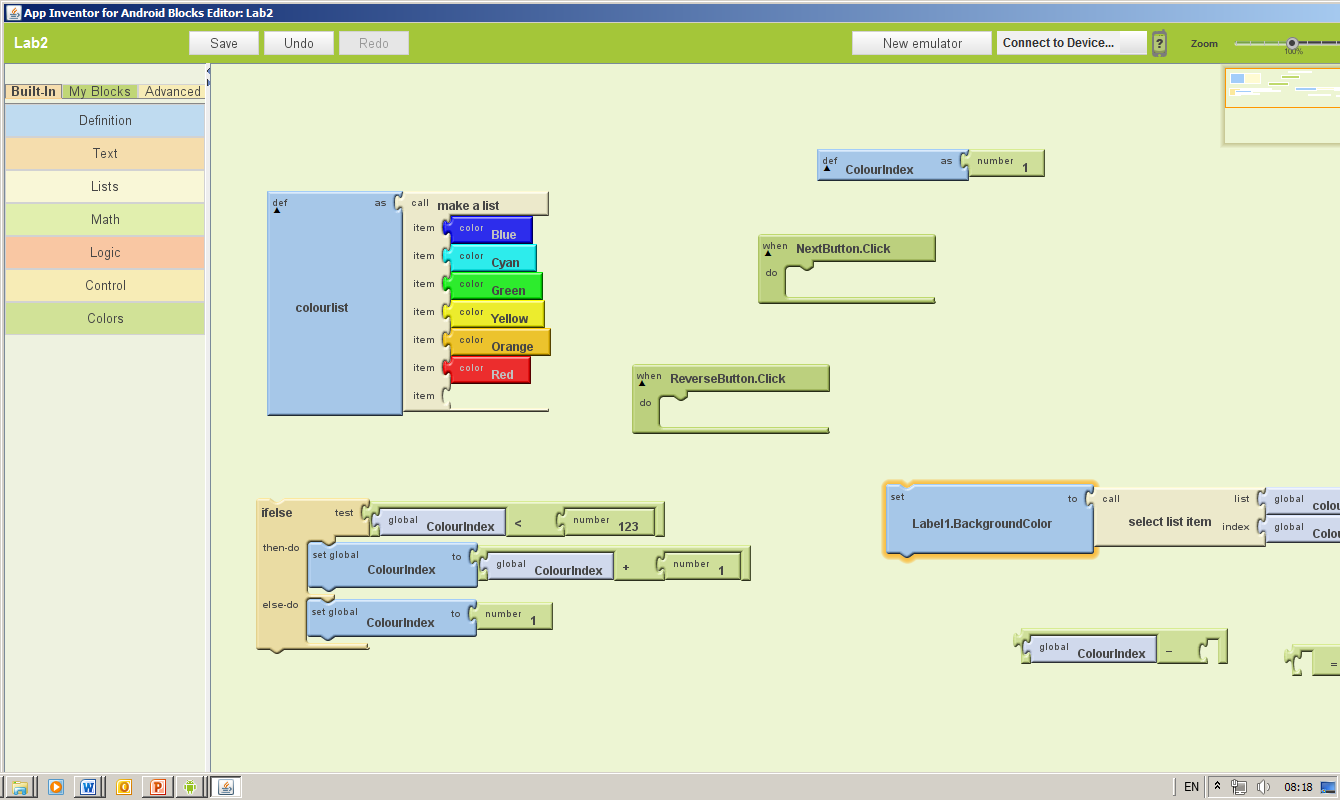
We will also need another variable to act as an index to the list, remembering the colour that we last used. The index will be a number (with a value between 1 and 6) which will correspond to the position in the list of the colour we want to remember.

Define a variable ColourIndex with the value 1.

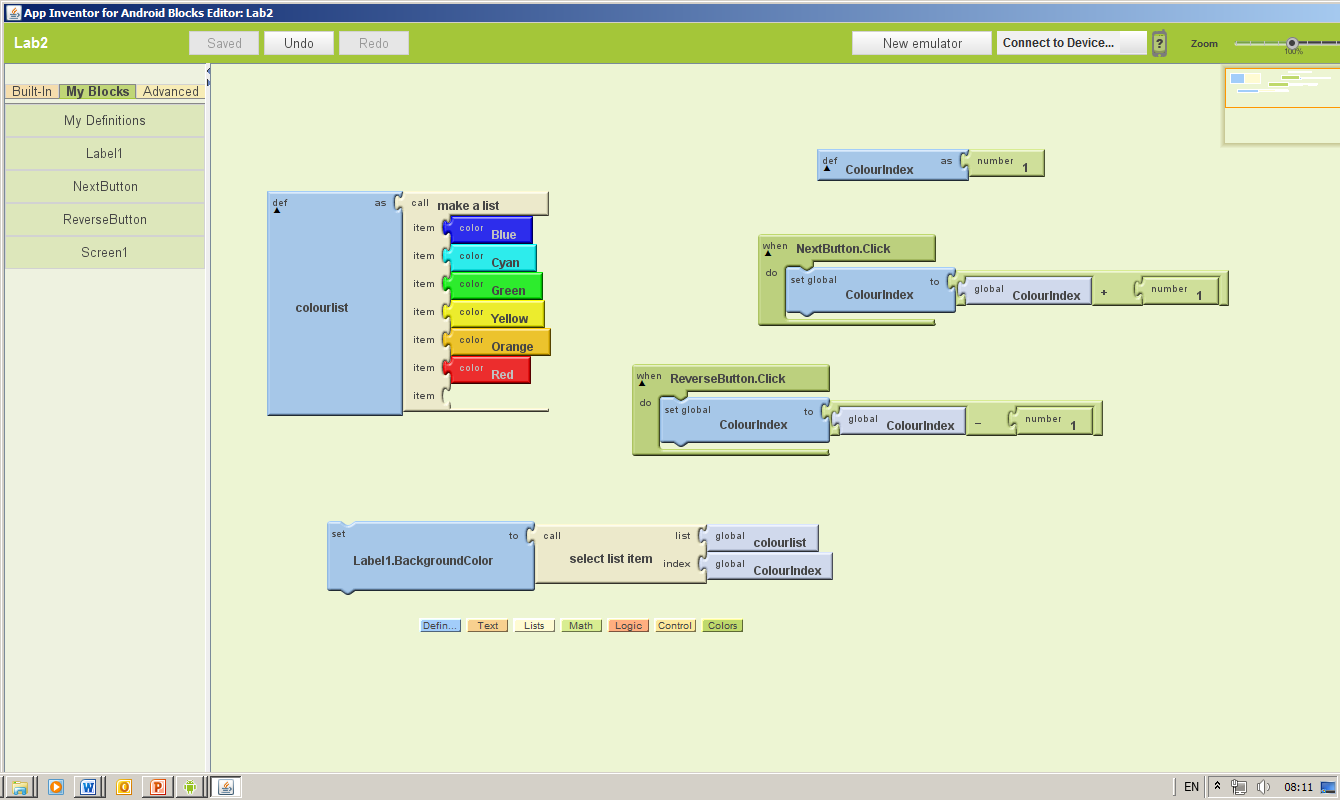
Now add an event handler for when the Next button is clicked. It has to do two things:

Firstly, increase the value of ColourIndex by 1. This is not quite so straightforward, because once the value goes above 6, it should wrap back round to 1 again.

We can manage this with an If-else block… If the value of the index is less than 6, add 1 to it, otherwise, set the value back to 1 again. Build the expressions as before, and insert them in the if-else block.



Secondly, we need to change the background colour of ColourLabel to the new ColourIndex-th colour in the ColourList. This latter is achieved by using the select list item block and plugging in the name of the list and the name of the index variable.



These two steps can be plugged together inside the event handler.

Run the app on the emulator, and watch the index variable.

The next step is to add the reverse functionality. I am going to give you much less guidance on this, and suggest you discuss your approach in twos or threes to try and work things out. Don’t be afraid to ask for help!

Define another variable called Forwards, and this time, give it the Boolean value of True.

Make an event handler for the Reverse button that changes the value of Forwards from True to False and from False to True. You could use If blocks, or better still you could use the not operator from the logic drawer.

In the event handler for the Next button, you now need to decide whether to increase or decrease the index. Be careful that the index doesn’t go out of range!

If you have got this far, ask a tutor to give you some points in return for you explaining to them how your app works.

Now work through the QuizMe tutorial at Google:  
<http://www.appinventorbeta.com/learn/tutorials/quizme/quizme.html>

This tutorial will show you one way to *begin* thinking about your own Quiz, and you could use it as a starting point. To make a multiple choice, you would need to have a list of questions, a list of *lists* of possible answers, and a list of correct choices.

To meet the minimum specification, your quiz should present the user with a number of questions, one by one, in a fixed order. The user should be able to select an answer, and then proceed to the next question. There needs to be some way of letting the user know their score at the end.

How you design the interface for your quiz is up to you. The tutorial suggests using the ListPicker component… you would need to look at Google’s component reference documentation on-line to learn how this works. An alternative (which would look more like a traditional quiz) would be to have buttons to select an answer.

There are many ways you could enhance this basic behaviour. Be inventive! Bur remember, it is better to have something simple that works than something impressively complicated that doesn’t work!