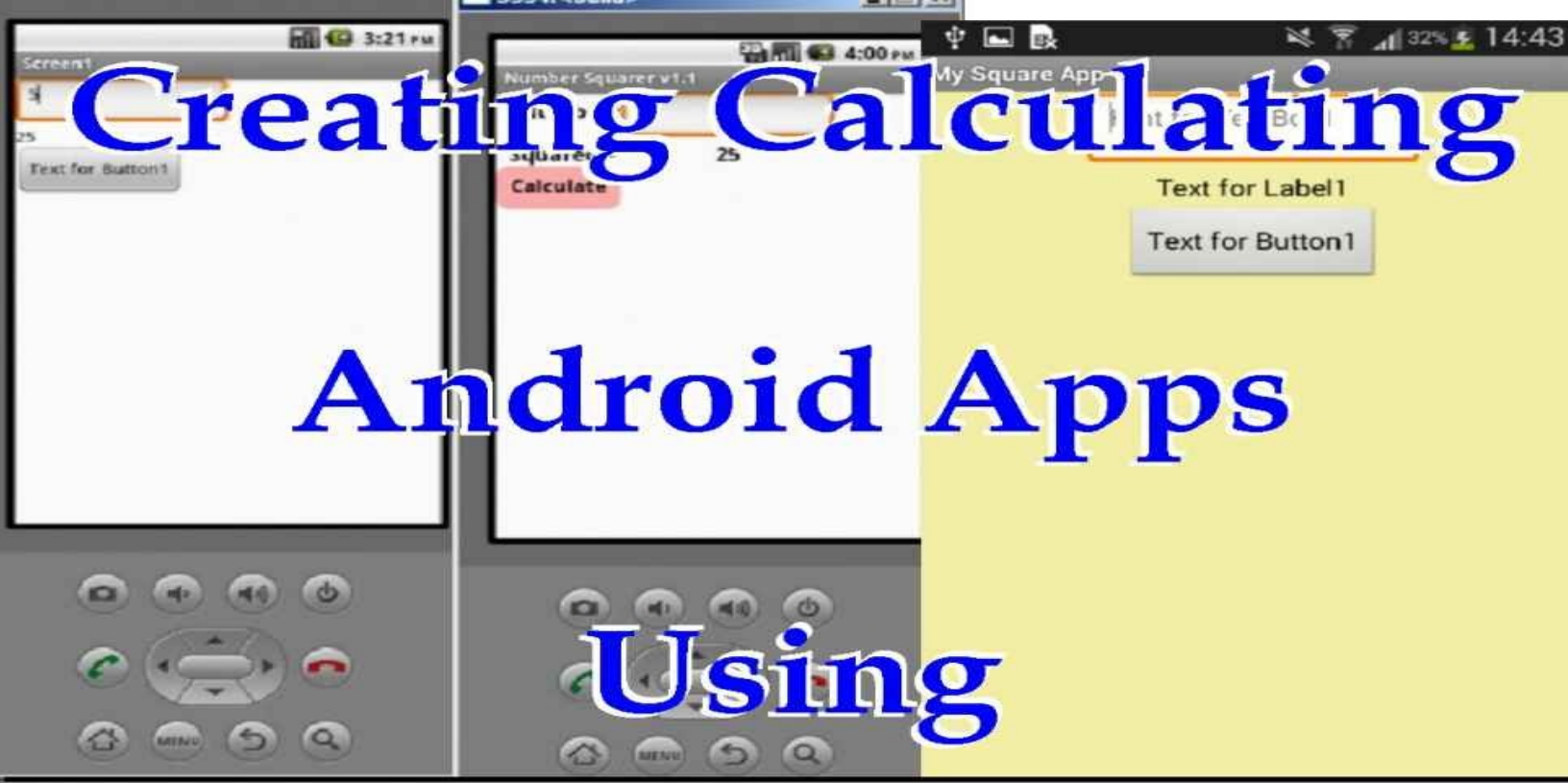
Creating Calculating Android Apps Using MIT App Inventor2



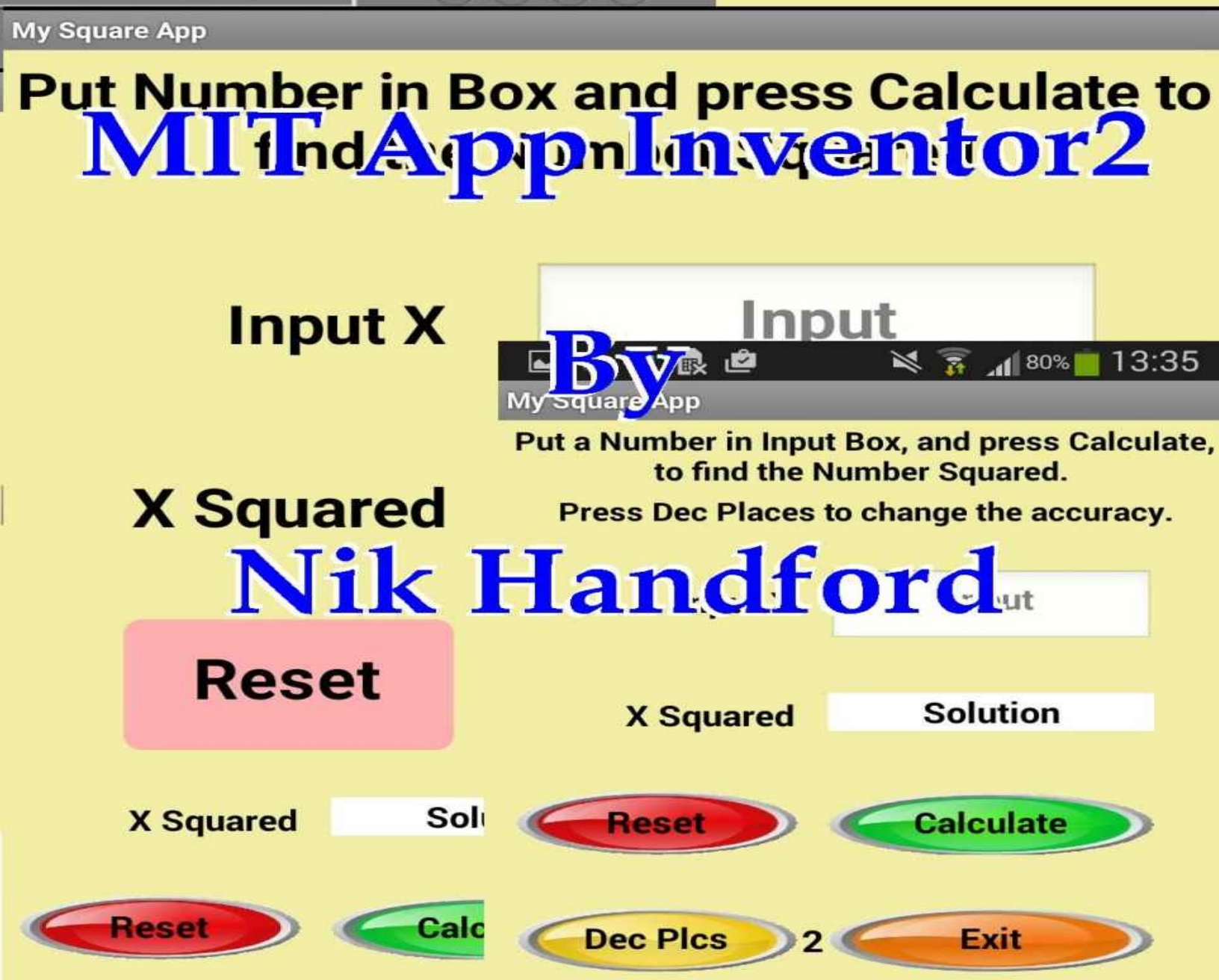


Table of Contents

Creating Calculating

Android Apps

using

MIT App Inventor 2

Edition 1.0 Jan 2017

By

Nik Handford

How to use the book

If you know all this, then you know how to skip this chapter.
If you're using a HD device and the pictures look small you can zoom them with a double tap. And even with an old device they can be zoomed, by selecting them.
To find your way around this book you have a number of options:

Start at the beginning and read through.

Press **Menu** and then **Go to...** and then **table of content**.

You can flick through from Chapter to Chapter by pressing the **Cursor/Nav Right** to skip forward or **Cursor/Nav Left** to skip backward.

The book is split into Chapters and some are further split down into sub chapters. Each of these sub Chapters are included in the **Table Of Content,** so you can go to the Menu and find them.

If you you can't find the thing your after in the Contents or want to find a word, subject etc. like tif (an image file format) just type in the word (in this case **tif**) and click find. Then the Kindle will display everywhere that **tif** appear in the book, in order, with a link to it and some of the text around it.

A number of Sections have links, some are internal (elsewhere in the book) and others external (on the internet) if you select one of these and you're not online it will ask **Turn Wireless On?** If you say yes and it connects it will take you to that site. If you don't want to just press **Back** twice and you'll back where you first clicked on the link.

If you follow a link and want to go back to the place you first clicked on it just press **Back** once. Hopefully this will aid your enjoyment of this book and others if you didn't know this before.

About this book

Welcome to this, the 1st Edition of this book.
Why did I write "Creating Calculating Android Apps using **MIT App Inventor 2**"?
Well I have written a number of books on how to do calculations, inputs and outputs, using Java and JavaScript(via Html) and since I started using **MIT's App Inventor 2** I thought it would be of interest to a lot of people who don't want to learn Java, JavaScript and/or other languages, honest no language to learn it's more graphical/visual as you create by putting buildings block together. The results are the same but you don't have to know why things work only that they do. This leaves you to figure what inputs you require and the calculations you need to generate the outputs.

What is **MIT App Inventor 2**, basically it's a website run by The Massachusetts Institute of Technology, which will allow you to create a screen(in Designer) or user interface and then allows you to build functions(in Blocks) into your App to do what you, the user wants. It can do many other things but this book is purely how to create Apps that Calculate answers given a number of Numerical inputs from a user.

These Calculations can be anything from one input for working out say how many frames or Gb required for **X** minutes of film. To maybe 12 inputs and lots of outputs, with the ability to edit any one of the inputs and recalculating or change the number of decimal places the answers come out to.
Yes you could do the same with a spreadsheet. But do you carry a spreadsheet around with you, generally no, but you have your Phone or a Tablet or many other Android Devices these day. It won't require a large program to run it, each Apps is usually less than 2Mb. And once installed you don't need to be connected to the internet to use them. Also unlike a spreadsheet you can give it to others and you or they can't accidentally change anything other than the inputs.

What would you use it for? Well I started programming years ago after a friend of mine spent more than a day working out some calculations and when he came up with an answer that didn't fit his requirements, he had to change one of the inputs and start all over again. Another day wasted and if that wasn't right another. Eventually he'd accept the best results(of the ones he worked out) but obviously he could only try so many different inputs before he ran out of time. I asked him for his Calculations and 18 hours later I had a SuperBasic(Sinclair QL) program that would take his 10 or so inputs and come out with a list of outputs almost as you hit the enter key. He was impressed and used it. As he did so I watched him and he kept changing one of the inputs, and I asked what he was looking for, he said he was after the smallest variation in one of the outputs. So I went away and wrote a looping program which looked at that output value and then reduced the input he'd been changing by an amount and kept looping until the output was under 1·0" it then reduced the input to a tenth of what it had been doing until the output was less than 0·1" and then that repeated until we got the out down to a under 0·001" value. This was in the eighties, on a Sinclair QL it took a few minutes but it did it, and had gone through enough calculations that it would have taken him weeks if not months or years to do.

Something I learned from this, is that often when you start doing something, especially for someone else, they and you think your looking for one thing but as you achieve that you find that there are more stages you can build on or build in that will aid the user. If it is for someone else it's important to see how they use it as often there are figures that they almost always use, that you can build in, so they don't have to input it every time. You can also do it so that it's editable, so if they did need to change it they could.

If you're working out the area of a garden for seeds your not going to require the answer to 3 decimal places. But using the same App you could be working out the area of something a few hundred thousands of an inch or millimetre and you may need the answer to 5 or 6 decimal places. So we can build in a option that will allow you to change that. You could have different Apps for different accuracy but what a waste it's going be, it's the same accuracy, only at the output(display) stage will it be rounded to the format you or the user require.

If the App is for you then it's easier because you'll know how it works and what you expect to put in and hopefully what outputs you require. If you are doing it for someone else or to sell, you need to get as many people as possible to try it and see how they use it. Chances are first person you get to try it will do something you never expected.

One thing you have to cater for is finger trouble, if someone puts in a number with 2 decimal points in it or a letter or even leaves one number out, you'll want to put in a routine for checking this and that tells the user they must put numbers in. If not your App will crash and do your reputation no good. It is a pain but if you want people to use your App then it's worth it. Even if it's for yourself it's worth making it as foolproof as possible.

You'll want to be able to tell the user this requires 2 inputs
or
The first number must be larger than xx

You can use these Apps to work out any mathematical equation/s you want it to:

At Work
Areas of circles, triangles, rectangles, etc.
Areas and Volumes of cylinders, spheres, prisms, etc.
Gear ratios
Cutting Speeds

At Home
Take 2 prices for 2 different amount of something and work out which is the cheaper.
Amounts of ingredients for a different number of portions, if that's what you need.
VAT on prices.

Hobbies
Scale sizes, speeds etc.

The differences on rolling height, speeds or overall gearing of 2 tyres.
Car, Bike spring rates.

And the list goes on. The only limit is your ability to find the equations you need to work something out. This can be less than straight forward. In the example I started with we had 10 inputs and a few pages of calculations none of which are that difficult, just a lot of them, and the results from one feeds in to another equation and that into another equation and then those results are displayed and compared to the originals.

System requirements

Computer and operating system

Macintosh (with Intel processor): Mac OS X 10.5 or higher

Windows: Windows XP, Windows Vista, Windows 7. Not mentioned but see below I have tried with Windows 10 and it seems to work.

GNU/Linux: Ubuntu 8 or higher, Debian 5 or higher (Note: GNU/Linux live development is only supported for WiFi connections between computer and Android device.)

I'm using XP on an old(pre 2008) PC and Laptop. I have tried Windows 10 and that appears to work.

Browser

Mozilla Firefox 3.6 or higher (Note: If you are using Firefox with the NoScript extension, you'll need to turn the extension off.)

Apple Safari 5.0 or higher

Google Chrome 4.0 or higher

Microsoft Internet Explorer is not supported. I assumed this goes for Edge also as it's not mentioned so I tried Edge in Windows 10 and it appeared to work, and although it did say there was a bug when I started, it produced an App which worked.
In Chrome on Windows 10 I got no Error and all appear to work, although it was only for a trial on a friend's Laptop, so it's at your own risk.

I'm using Firefox on my laptop and Chrome(49.0.2623.112 m) or Firefox(49.0.2 - 50.1.0) on my PC both seem to work equally well.

Phone or Tablet (or use the on-screen emulator)

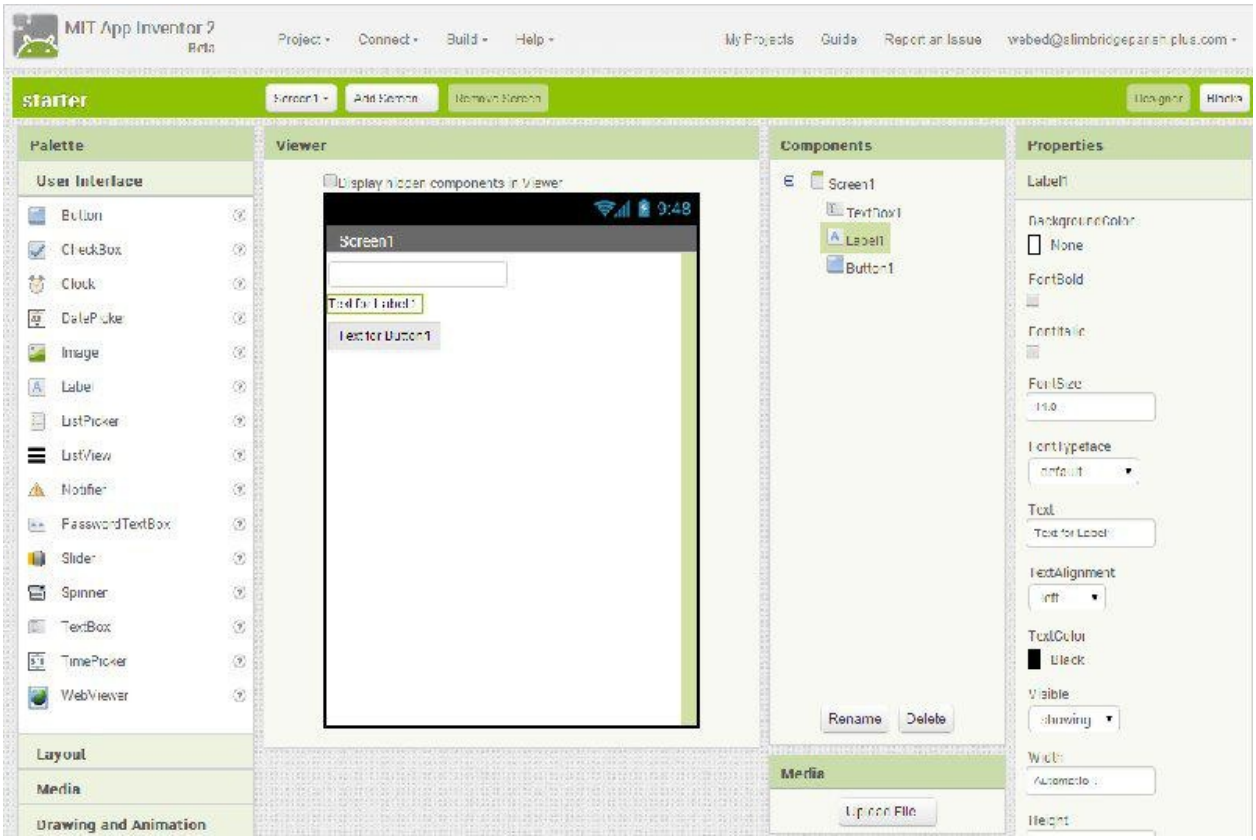
Android Operating System 2.3 ("Gingerbread") or higher

I'm using Marshmallow(6.0) on my E3, KitKat(5.1) on my Hudl2, 4.4.2 on my phone and 4.0.4 on my old table and all work with these **Apps**. I've also tried them on a friend's Kindle Fire and that too worked, which uses Amazon's version of Android.

Creating Apps

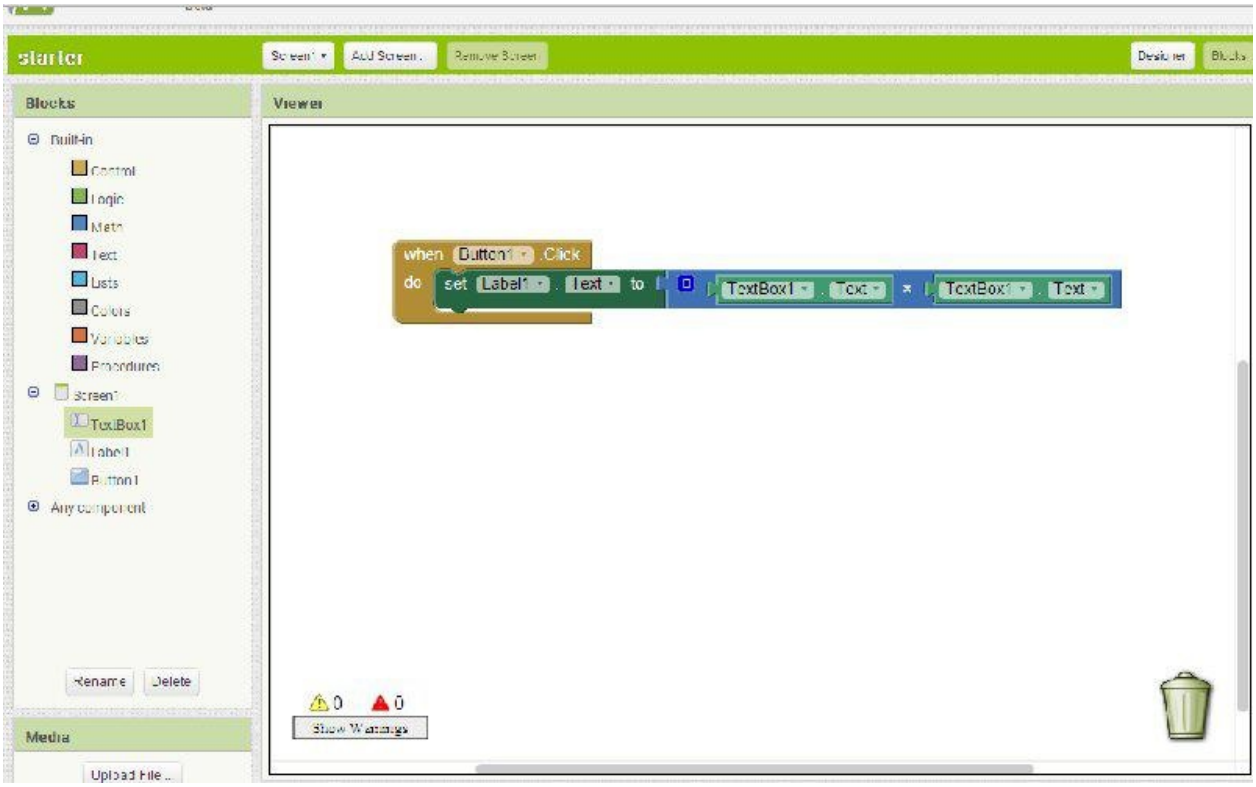
This may sound like it's for Advanced users, but it's really very easy. First a quick run through, just read it and we'll go through it thoroughly after.
On a PC or Lap-top you'll go to http://ai2.appinventor.mit.edu/ you can set-up an account (this is free) using your Google account or you can create a new one. This is the home of **MIT's App Inventor 2** it's very simple to use and it's got lots of help.

Basically you'll drag and drop a number of elements (buttons, labels, text-boxes etc.), in the "**Designer**" Screen, from your **Palette** to your **Viewer** window.

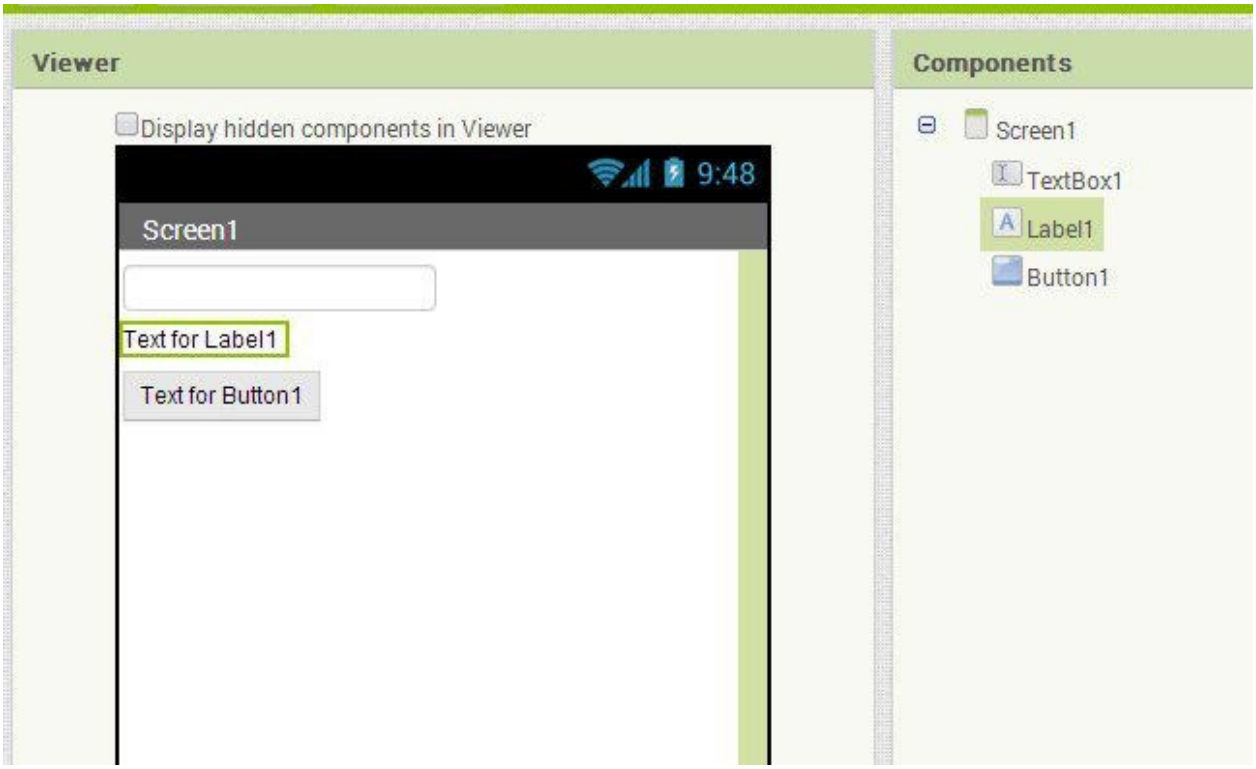


When it looks like you have everything you need you then go to the "**Blocks**" Screen.

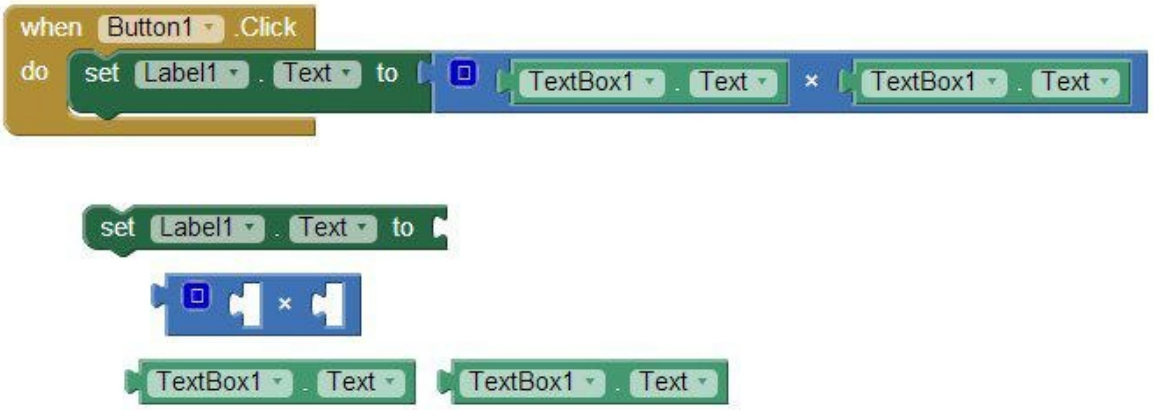
This is the complicated bit, you select the thing you want to operate (say the button) and it will give you a number of choices (like when button click do..). You drag this to the viewer side of the screen and then find the next stage you want it to carry out when you click the button and so on until you have an App.



A Quick Example



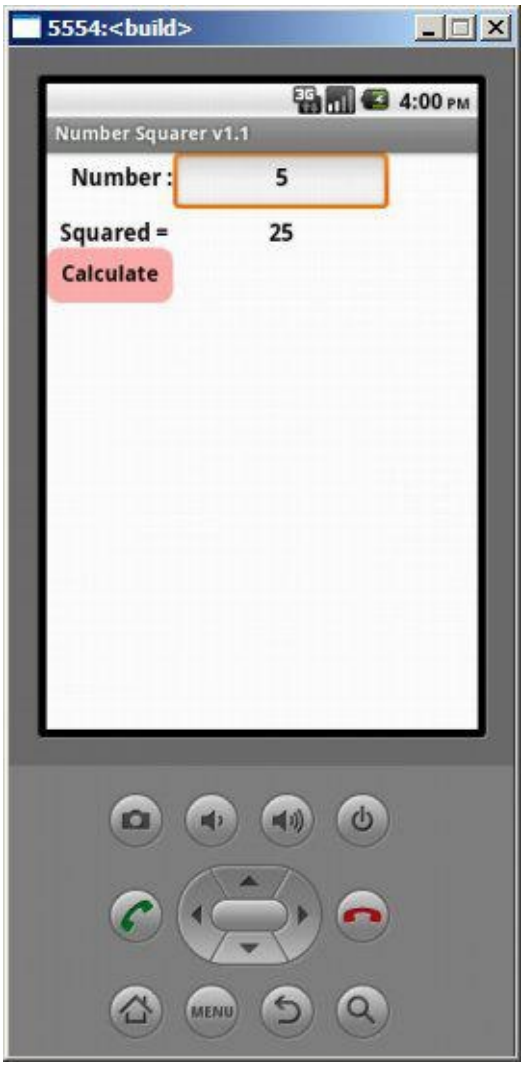
What I've done here, in the "**Designer**" Screen, is dragged over from the **Palette** to the **Viewer** window, a TextBox, Label and Button as in the example. Then gone to **Blocks** Screen.



Here, in the **Blocks** screen, I've gone to the **Blocks** window, selected **when Button.Click do** tag, dragged it over to the **Viewer** window. Then done the same for **set Label1.Text**, a **Multiplication** from the **Math block** and **TextBox1.Text**. Then in the **Viewer** window put them together like a puzzle and Right Click the **TextBox1.Text** and selected **Duplicate**.
Now if we go to our tablet and run it you can put a number in the **TextBox** press the button and it will square the number and display the answer in the **Label** area.
Obviously this is very basic but you can rename and format the screen and it's parts to do lots of things.

This is what that would look like on their emulator, pretty lame, but then I've only used 3 elements.

So you can add some more Labels/text and change the font sizes, justification and mess with the Button.



It looks a bit better and you can add images, colours etc. You'll get the idea and once we get you started, you can add more and make it do loads.
I.e. input a Diameter and work out the Area, and Circumference of the circle and the Volume and Surface Area of it's Sphere.

If you download "MITAI2Companion.apk" or get it on your Tablet/Phone from "Google **Play Store**", you can try a demo run and change it on your Computer and keep messing with it until it does what you want and looks how you want it to. When you're happy with it you can download the .apk file and put it on your tablet/phone or give it away or even sell it.

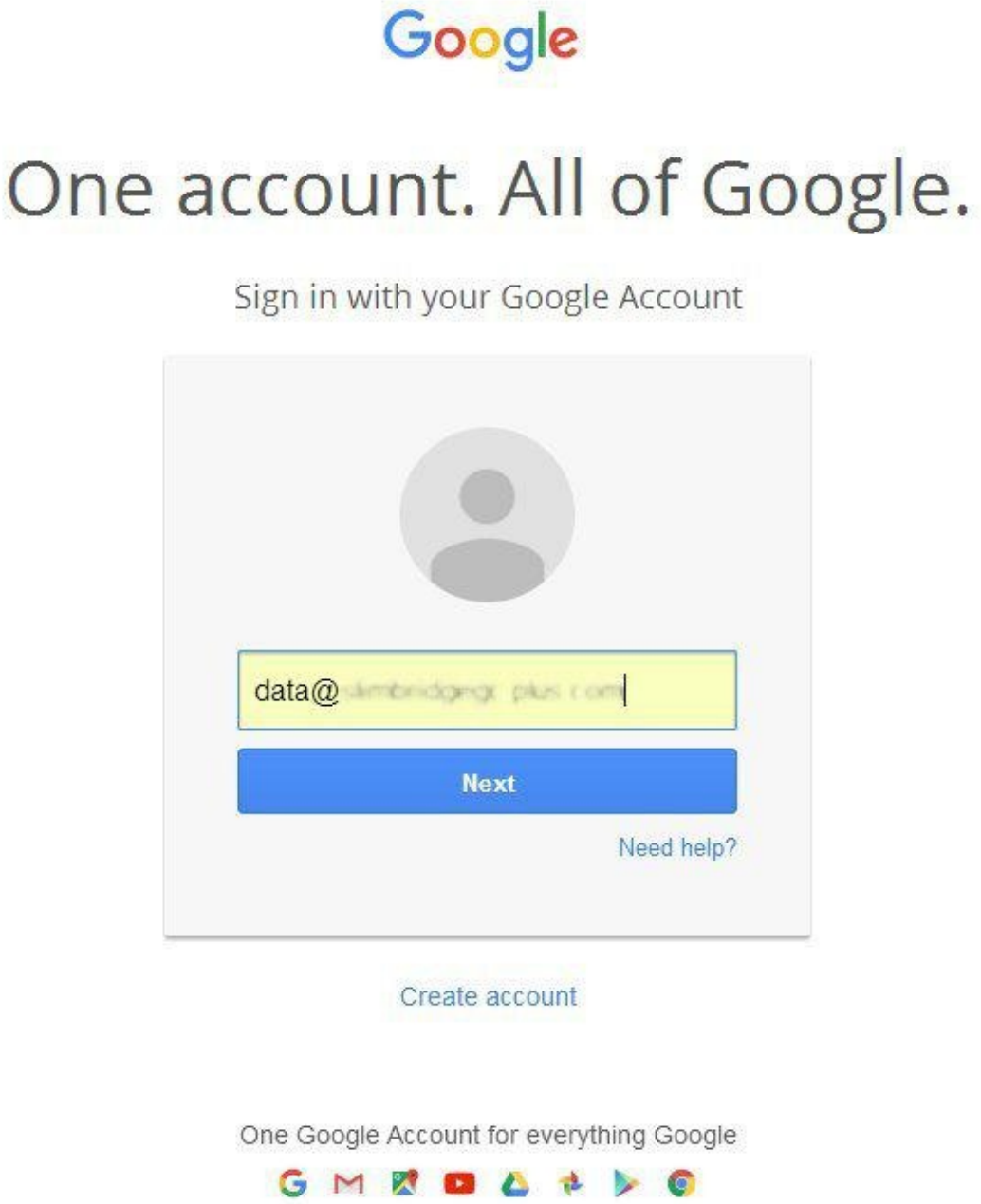
In this book I will only explain how to use **MIT App Inventor 2** to work out equations, but it has lots of other functions I've used like GPS, Accelerometer, Barcode Reader, Nearfield and Orientation Sensors. I won't be covering these but once you can do the Calculating Apps just find an example similar to what you want to do and you can see how to do these things. There are lots of examples you can download and edit to use as a starter to get yours going.
Have a go it's easier than you might think and certainly easier than it is to explain.
One word of warning, regularly download the .aia back up of your files as MIT take no responsibility for them, if they have a problem on their servers, your work may be lost. I have to say I have been using this for some time and never lost one file but if you want to be safe back it

up. The .aia files are editable, if you only have the .apk files you can load it, run it, but not edit your App.

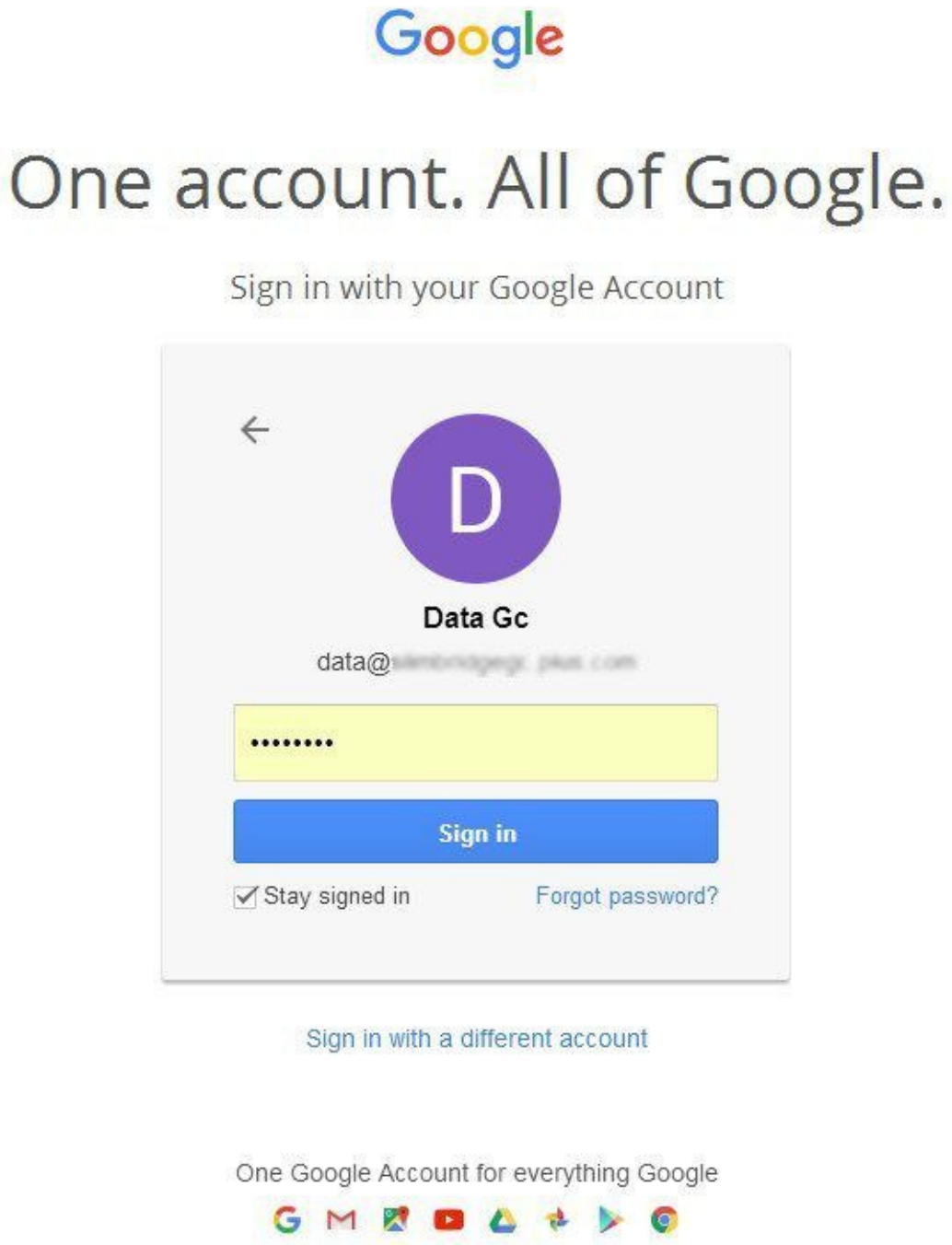
Our starter for 10

Sorry English humour
We'll start fairly basic as once you've learnt the basics it is easy to progress. Unlike most I hate the "Hello World" first step and so we start with an App that actually does something, that you can edit and make it do what you want instead of just displaying "Hello World".

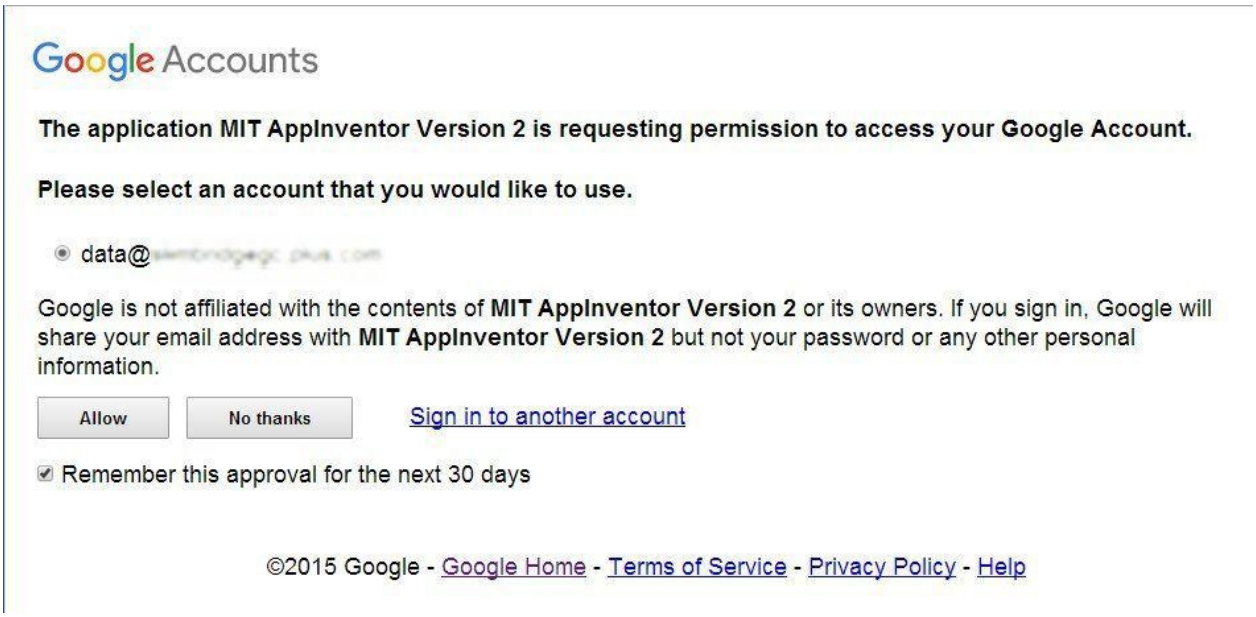
Let start by creating the App that was shown previously, it's very simple but will teach where everything is and how it works. It will take the user's input and work out the square of that.
Open your browser on your PC or laptop and goto http://ai2.appinventor.mit.edu/ I use Chrome and Firefox for this but it will work in others(see System Requirements), not IE.



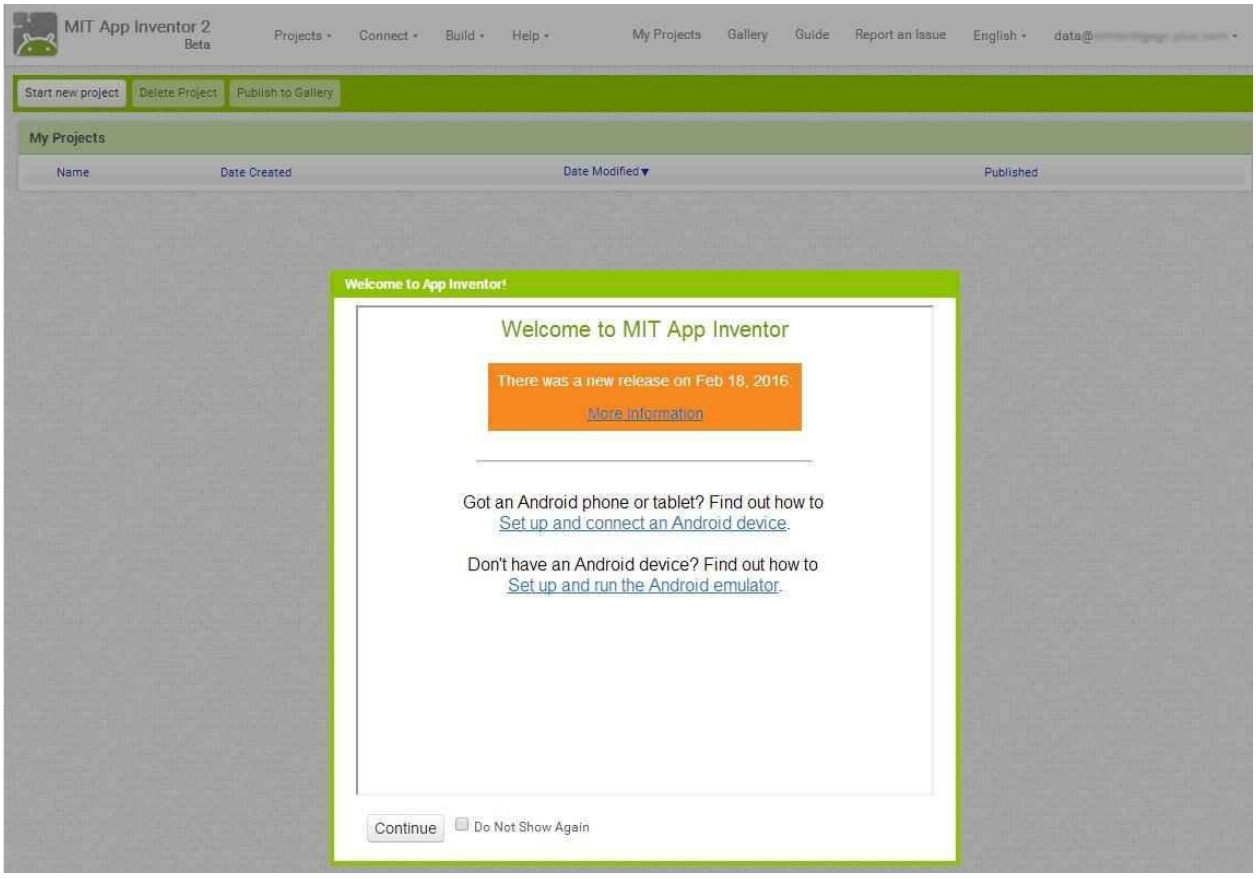
If your Google Username doesn't come up enter it here. And click "**Next**".



Enter your Google Password
You can select "**Stay signed in**" or not, I select it but it's up to you.
And Click "**Sign in**".

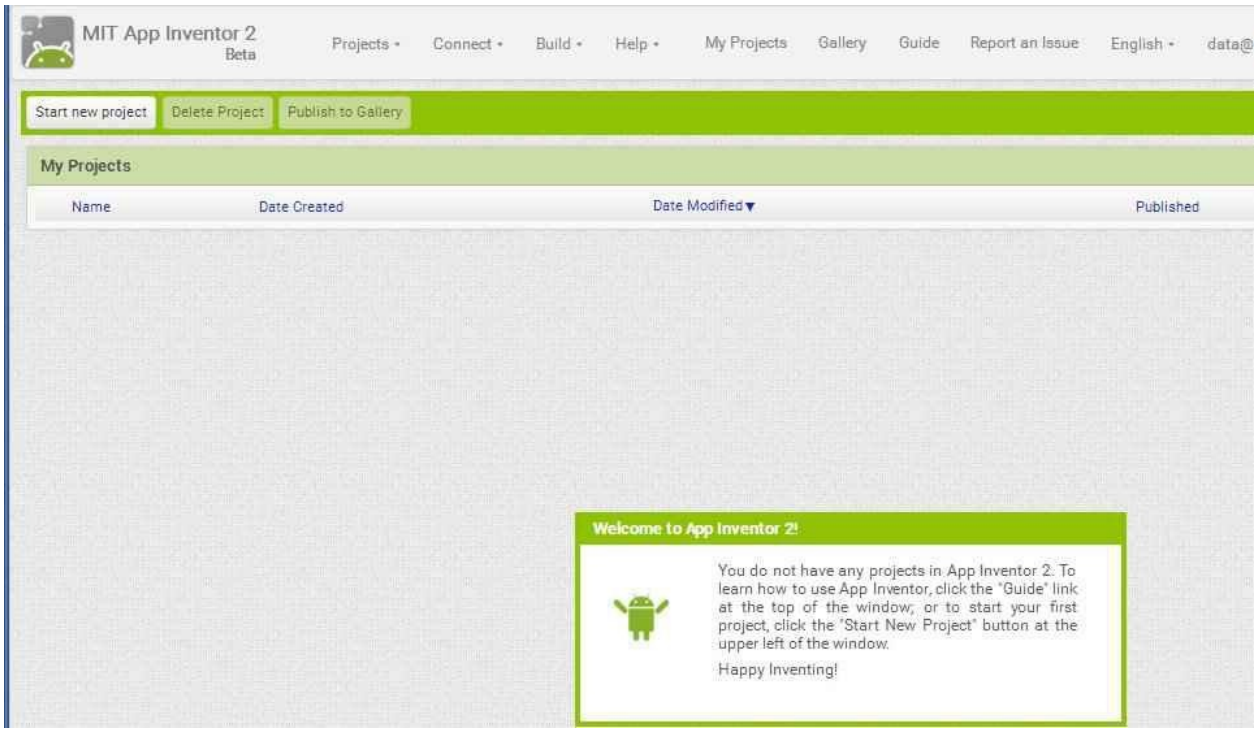


If you want to continue select "**Allow**". If you select "**No Thanks**" you will proceed no further. You will now see the Term of Service again if you want to continue select **I accept the terms of service** you will be allowed to continue. See Term of Service for the conditions (correct at time of publishing July 2016) or wait till you sign in.
You may be asked to take part in a Survey, please do, it will take a minute, if that.

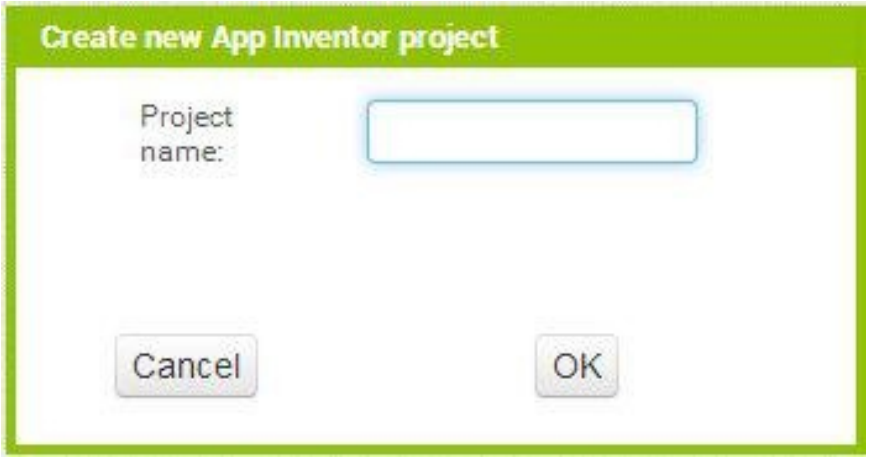


Nearly there. This is asking if we want to set up our phone or an Android Emulator. You don't need to do it now but if you want to you can. The best way is to install the **App Inventor**

Companion App on your device (see Installing The App).
OK you're here and ready to start writing Apps.



Select "**Start new project**"



Type in your Project's Name(Example_1) and hit **OK**. This is not your App name so don't worry about that yet. So you might have noticed I used an underscore where there would have been a space, you cannot use spaces in the name of your Project, and they must start with a letter not a number.

Before we Start

Ok before we create the App, we need to know a few basics:
What do we want it to do?
How many Inputs will it need?
How many Outputs will it need?
How many Commands/Buttons will it need?
What calculations do we need to do this?

As said before, we'll are start with the previous example, not exciting I know, but you will be able to use it and modify it as we go. So we start simple and that way if it doesn't work it's easy to spot the mistake and put it right. If you try doing a lot of calculations in one go it can take an age to track down any errors so I start with one calculation/process, run it, if or when that works put in the next and check it. If that works add the next and so on until you have all of your calculations and they all work.

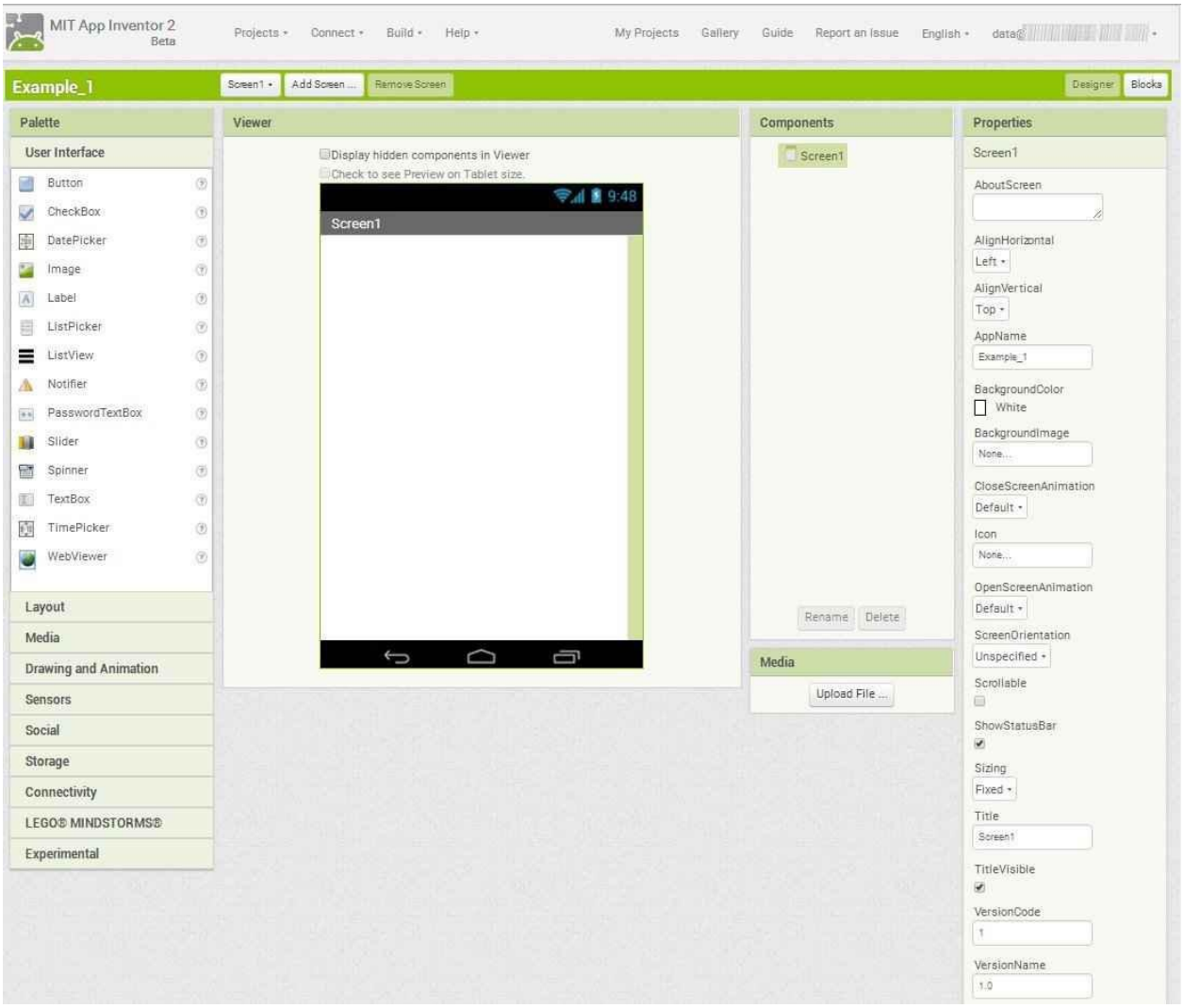
For this example we'll create an App that takes one number, input by the user and then, on command, times it by itself and display the result.
So we will need 1 Input.
And 1 Output.
We need 1 Command/Button. Calculate.
We need one calculation - input 1 × input 1 = output 1

That will be it for the first example but after we can edit and add to it.
We can check that the input is a number and not letters or anything else.
If it is not a real number then we can display a message to say "You need to input a NUMBER".
We can allow the user to change the output from 0 to 4 decimal places in the answer.

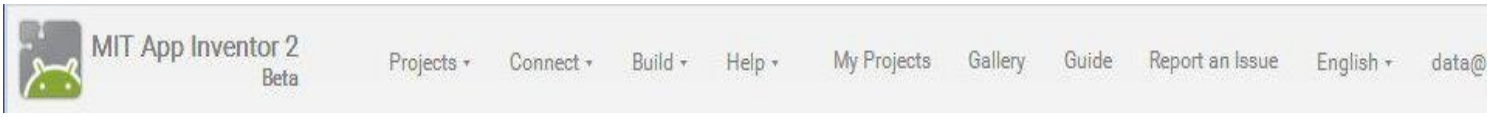
We will also create a reset button that will reset the input and output.

Example_1

If you're logged on to "**MIT App Inventor**" your Computer should look like this.



Starting at the top, this is the **Command Bar**.



But we won't worry about it at present, but will come back to it as and when we need to use it.

Under the **Command Bar** left hand side you'll find the "Project" name(Example_1), don't worry this can be anything, you can name the App anything you like, it doesn't have to be the same as the Project name.

Further along is "**Screen1**", "**Add Screen**" and "**Remove Screen**" don't worry about these we will be using "**Screen1**" only, but be aware if you want to have a "**Screen2**" or "**Screen3**" you can but we won't be covering it in this book.

Look to the right hand end of this row and you'll see we are in the "**Designer**" Screen. The Screen you are on is greyed(or should that be greened) out 

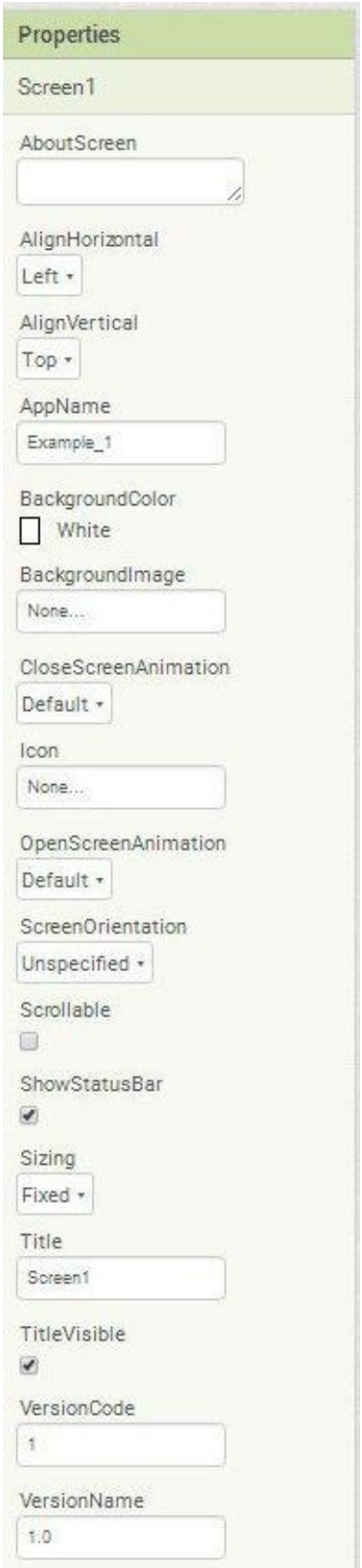
so you can click "**Blocks**" when you want to go to the Block Screen and when you're there "**Blocks**" is greyed out 

so you can click "**Designer**" to get back.
"**Designer**" is where you create the look, what the end Users will see on their Devices, "**Blocks**" is where the work is done, this is where it says this button does this, and where your equations are defined.

Below that you'll notice Four Areas, **Palette**, **Viewer**, **Components** and **Properties**.

First we'll be using **Properties** these are the properties of the highlighted **Components**, in this case **Screen1**, as it's the only thing in the Project so far.

Properties


Properties
Screen1
AboutScreen
AlignHorizontal
Left
AlignVertical
Top
AppName
Example_1
BackgroundColor
White
BackgroundImage
None...
CloseScreenAnimation
Default
Icon
None...
OpenScreenAnimation
Default
ScreenOrientation
Unspecified
Scrollable
ShowStatusBar
Sizing
Fixed
Title
Screen1
TitleVisible
VersionCode
1
VersionName
1.0

AboutScreen

Information about that screen/page. It appears when the user selects "**About this Application**" from the system menu. Use it to tell users about your App. If you have more than one screen, each screen will have its own AboutScreen.
i.e. Square Calculator - This App will work out the Square of a number.

AlignHorizontal

Do you want you main screen horizontally aligned Left, Center or Right.
Center sounds ok but we can change if it looks wrong.

AlignVertical

Do you want you main screen vertically aligned Top, Center or Bottom. This will have no effect if the screen is **Scrollable**.
We can stick with "Top"

AppName

This is the display name of your App when installed on a Device. If the AppName is is not filled in the default will be set to the project name when the project is built.
In this case **Example_1** but I will rename it **Square App**

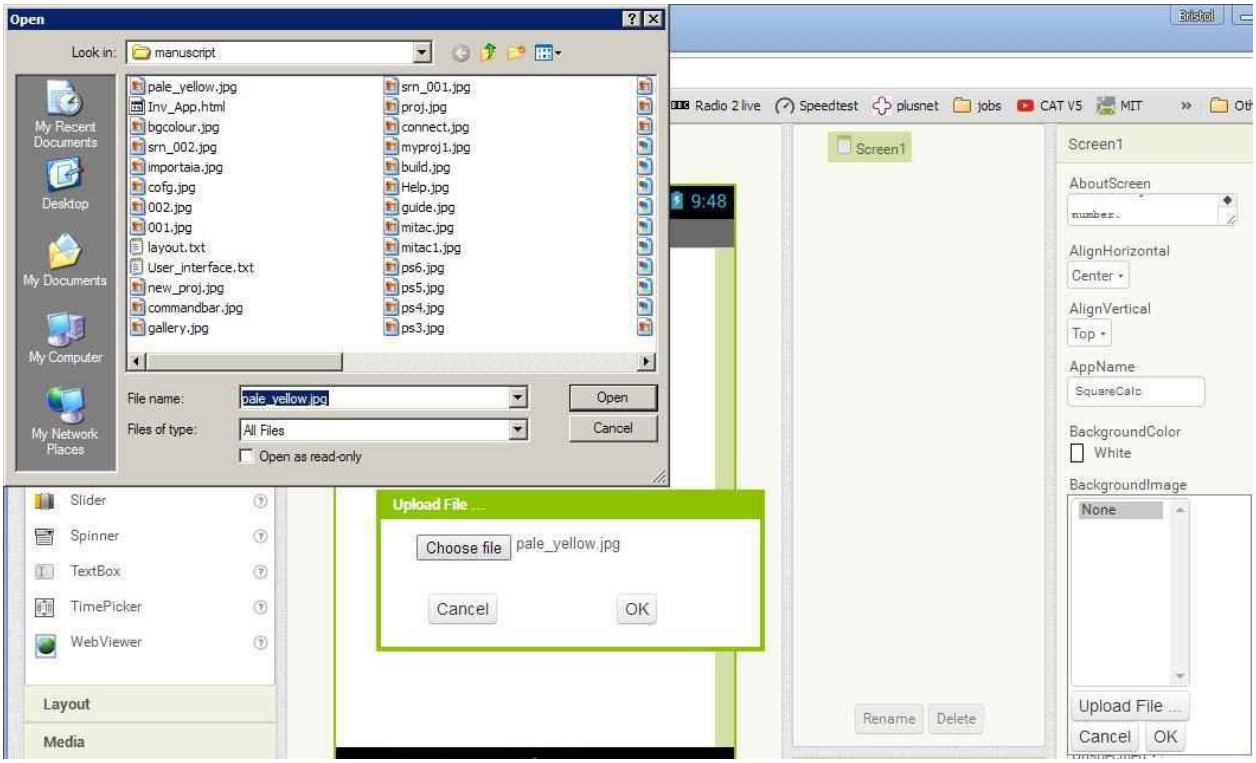
BackgroundColor

Pick a colour, any colour from the following.

If you don't like any of them use **BackgroundImage**

BackgroundImage

Use an Image as your screen background.



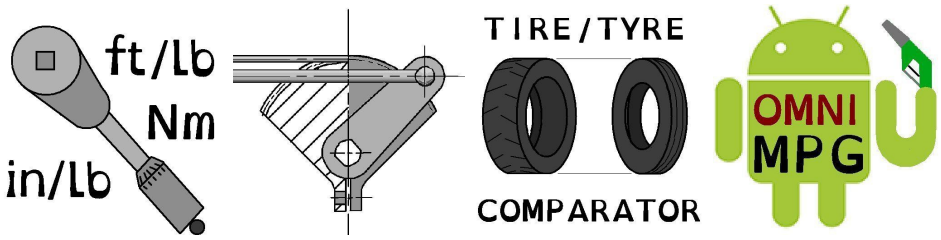
If you don't want to use an Image and didn't like any of the 16 colours, you can create a 50x50 .jpg any colour you like and get the background colour you want. Remember if you use an Image it may change the look from Device to Device because of the Resolution and you may have to force the user to use it in Portrait or Landscape to make it look right.
If, like me, you create a colour .jpg to use:
Click the **None...** below **BackgroundImage** and then click on **Upload File...**
Then select choose file, use the window to find the image you want, select it and click "**Open**". The file Name should now be in the "**Upload File...**" box, if it is, click "**OK**", as you do this the Device Window in the **Viewer** area should show the image.

CloseScreenAnimation

Leave

Icon

Leave this for now but be aware if you want each of your Apps to have their own Icons, bookmarks on your home screen then you will need to think up some unique Icons 512 pixels by 512 in .png and .jpg format and one 114 pixels by 144 as a .png. I use the .jpg in the App and the other 2 are if you want to put it on Amazon, free or for sale. Anyway for now will miss it out.



These are some of the ones I've created.

OpenScreenAnimation

Leave

ScreenOrientation

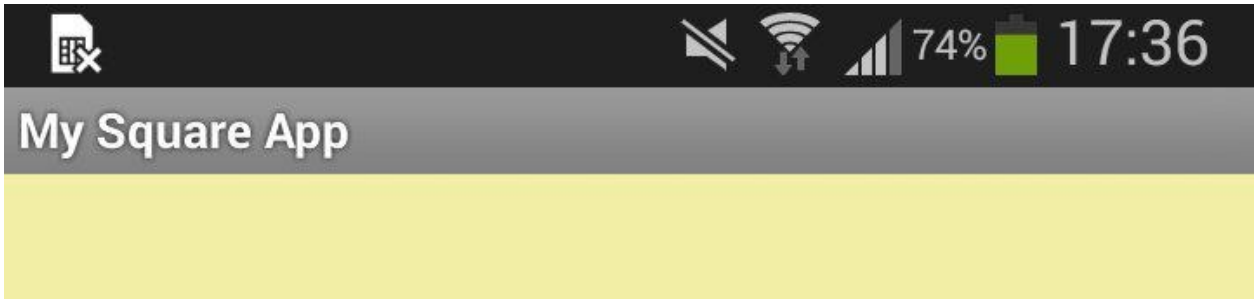
The screen orientation. Can be sensor, user or unspecified.
If you specify Landscape or portrait then it will always display that way round on a device and only that way. This might be useful if your App only looks right one way around in which case use Landscape or Portrait if not go with User. This will probably be especially useful if you're using a background image.

Scrollable

When checked, there will be a vertical scrollbar on the screen, and the height of the application can exceed the physical height of the device. When unchecked, the application height is constrained to the height of the device.

ShowStatusBar

Unticked - no Status bar will be shown on your Device.



Ticked - the status bar will be shown.

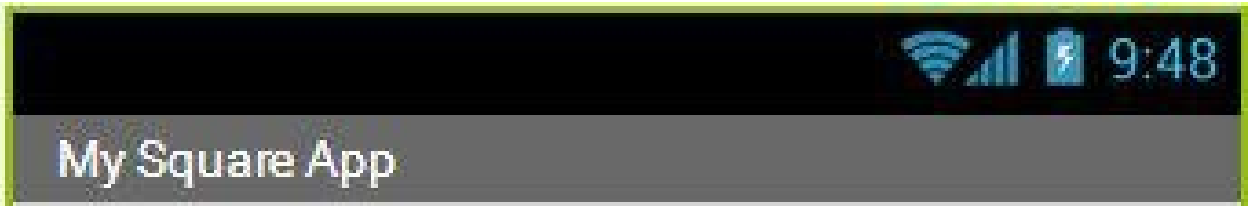
It will always appear on the Preview screen, I don't know why.

Sizing

If set to fixed, screen layouts will be created for a single fixed-size screen and autoscale.
If set to responsive, screen layouts will use the actual resolution of the device. See the documentation on responsive design in App Inventor for more information. This property appears on Screen1 only and controls the sizing for all screens in the App.

Title

The caption for the App, which appears in the Grey title bar



Here I called it "My Square App"

TitleVisible

The title bar is the top grey bar on the screen. This property reports whether the title bar is visible.

VersionCode

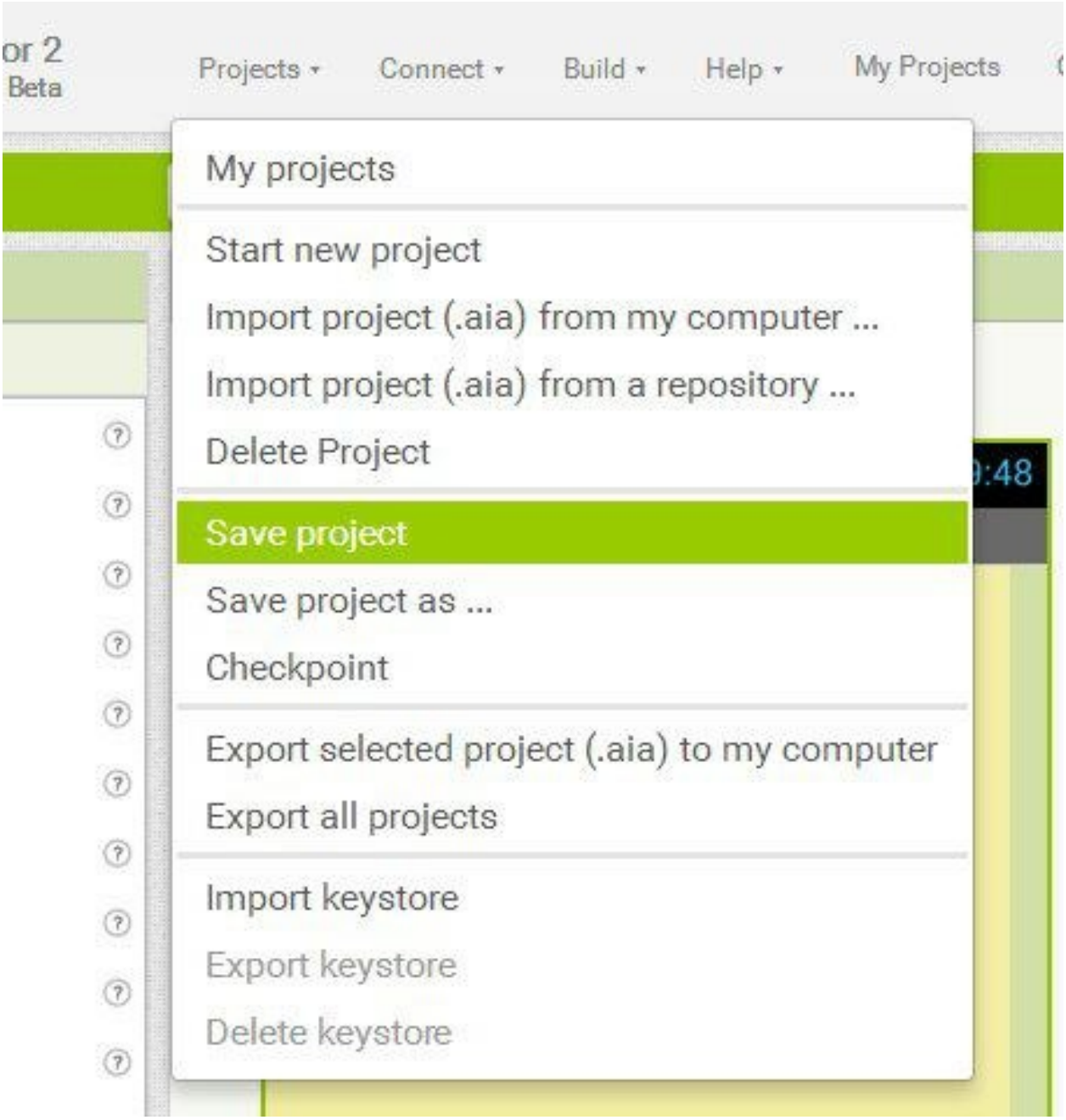
An integer value which must be incremented each time a new Android Application Package File (APK) is created for the Google Play Store.
I start with **10** and then **11** if it's a small change or **20** if it's major.

VersionName

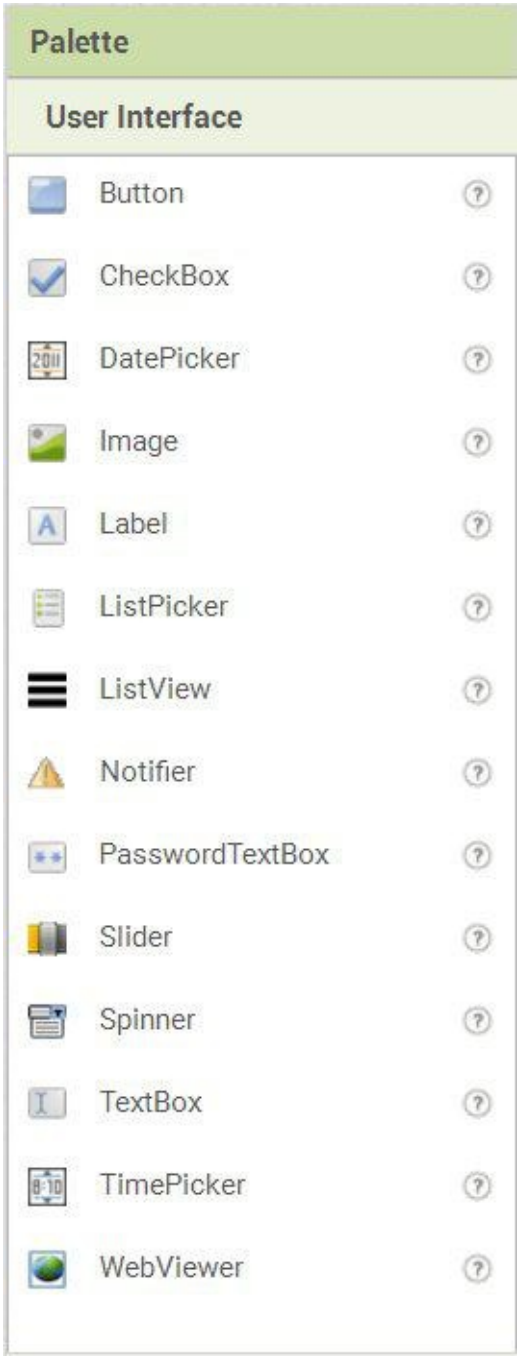
A string which can be changed to allow Google Play Store users to distinguish between different versions of the App.
In the previous cases it would be **1.0** and then **1.1** or **2.0**.

If my VersionCode is **15** then my VersionName will be **1.5**, you don't have to follow this way of working but it works for me.

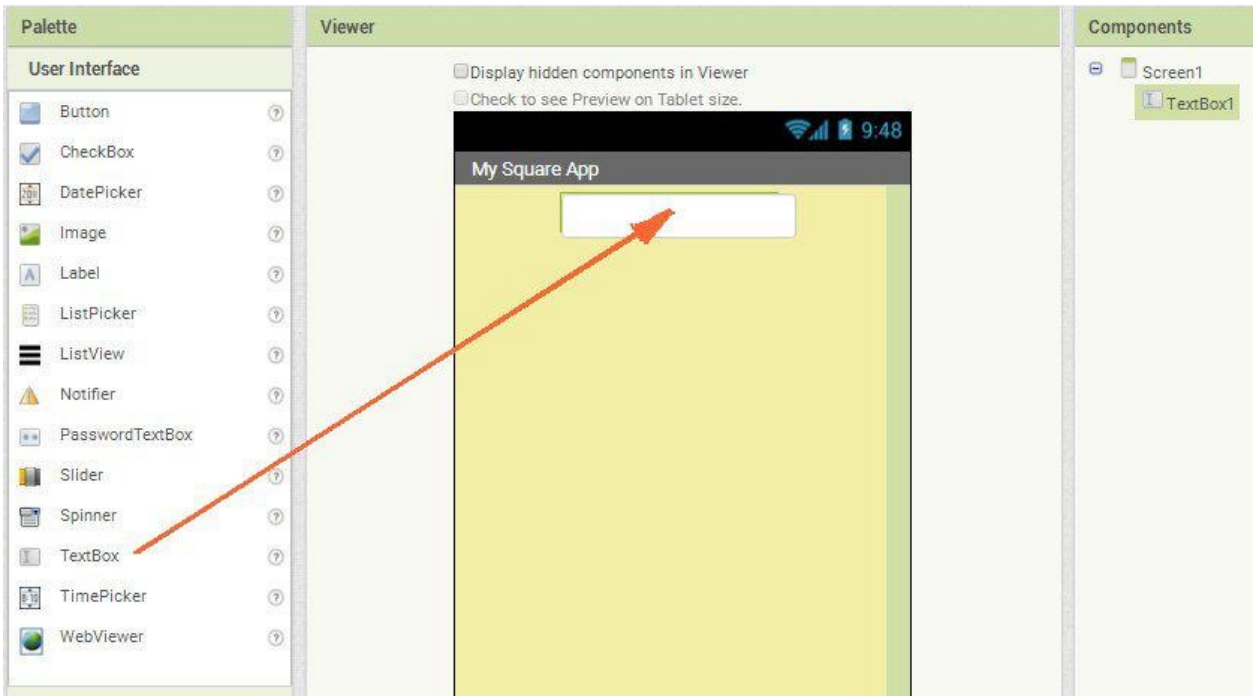
Ok that's Properties for Screen1 done the important thing to remember is that nothing you have done is set in stone you can change it anytime you want to.
Now click on **Projects** in the **Command Bar** and then **Save Project**



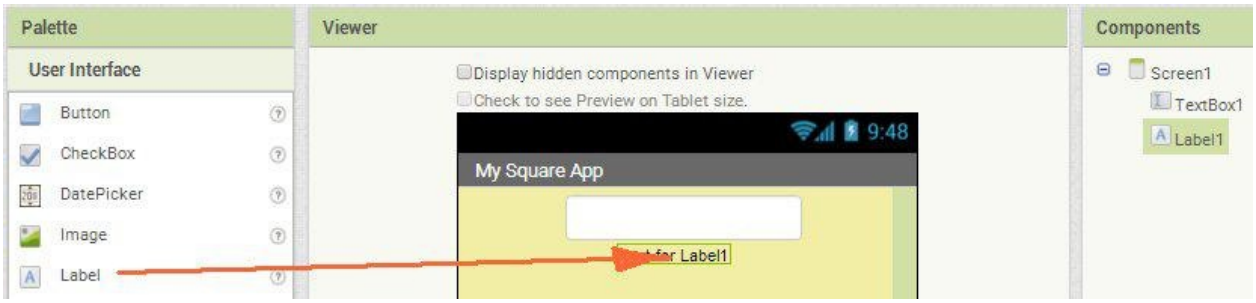
Time to use **Palette** with **Viewer**. We need an Input box, a Calculate Button and an Output window.
The Input will be done using a **TextBox**
The Button will be done using a **Button**, you probably didn't see that coming.
The Output will be done using a **Label**
These can all be found in **Palette**, **User Interface**.



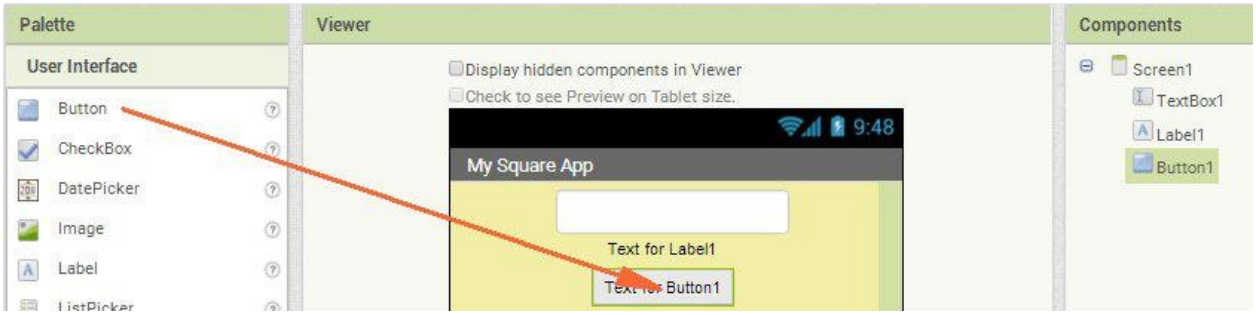
We need an input Window, in this case we use a **TextBox** from the **User Interface** and we Click, Drag and drop it into **Viewer**.



Next we need an output Window, in this case a **Label** from the **User Interface** and we Click, Drag and drop it into **Viewer**.



And we need some way of make it do what we want, a **Button** from the **User Interface** and we Click, Drag and drop it into **Viewer**.

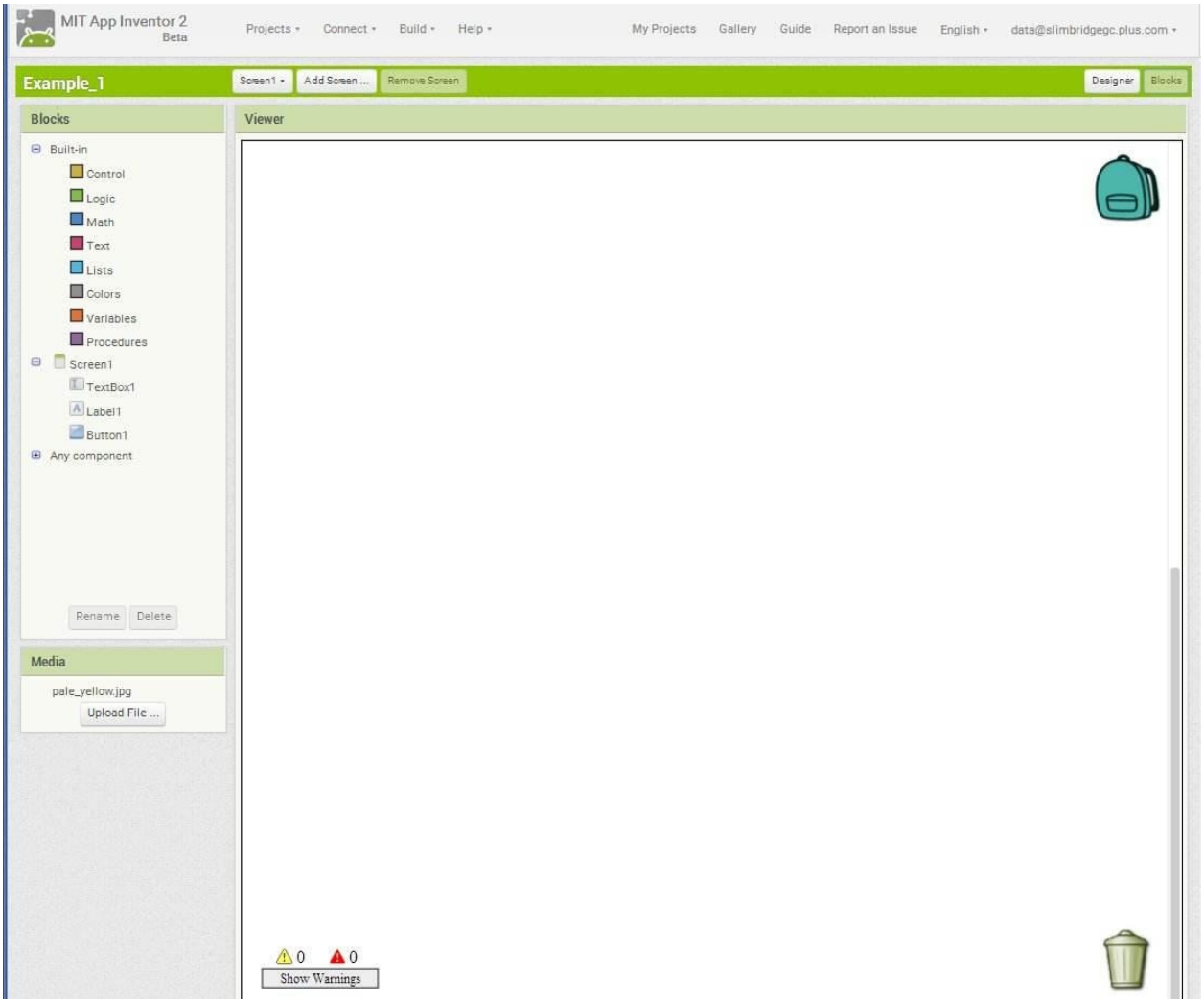


You should have noticed that in the "**Components**" Window you now have "**TextBox1**", "**Label1**" and "**Button1**". As you drag things across to the **Viewer** window their IDs will appear in the **Components** Window. They would be given a unique name so the next Textbox created will be "**TextBox2**" then "**TextBox3**" etc. and the same with the other **Components**. These names will be confusing later when you have a few of each so we will rename them but for this part in the

exercise we will leave them as they are.

Now Click the "**Blocks**" Button, top-right.

You are now in the Block Screen, which comprises of "**Blocks**", "**Media**" and "**Viewer**" Windows



"**Blocks**" Contains:

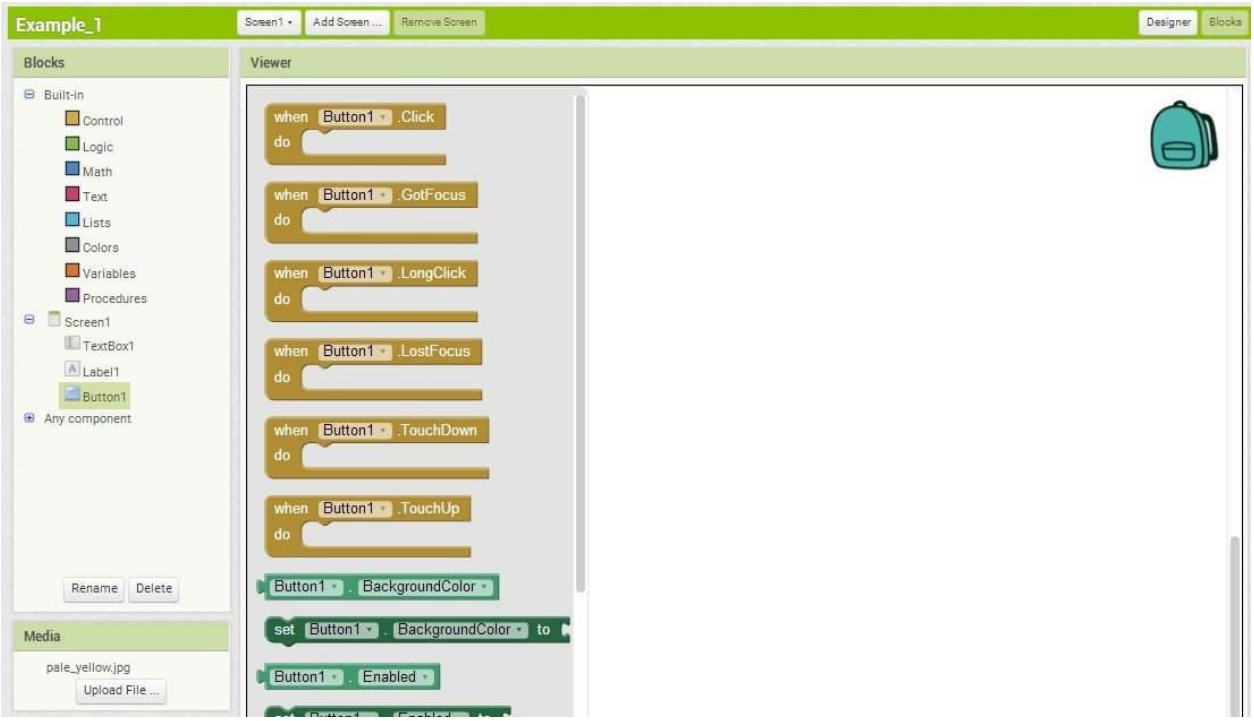
"**Built-in**" - **Control**, **Logic**, **Math**, **Text**, **Lists**, **Colors**, **Variables** and **Procedures**. These are the commands and instructions that we will be using to run/control our App.

"**Screen1**" - your components used in Screen1, had we used a Screen2 we also it and it's

components here. In this case under Screen1 we see our components **TextBox1**, **Label1** and **Button1**.

"**Any Component**" - We may come back to this?

Select **Button1** from the **Blocks** Window

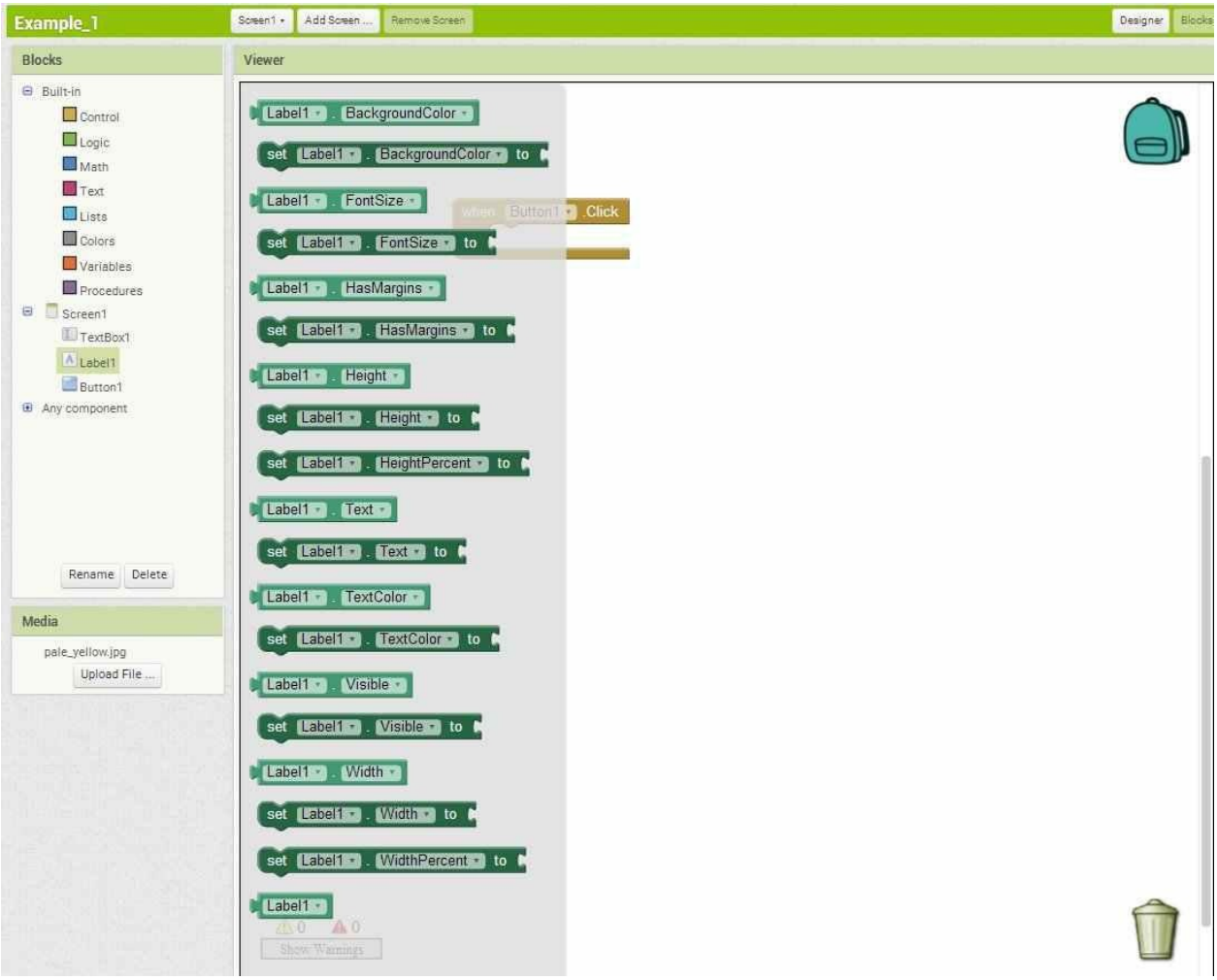


Then Click, Drag and Drop 

"**When Button1 .Click do**" to the **Viewer** Window.

So what have we done? We've said when the button, "**Button1**" is touched/clicked, do something.

Select **Label1** from the **Blocks** Window



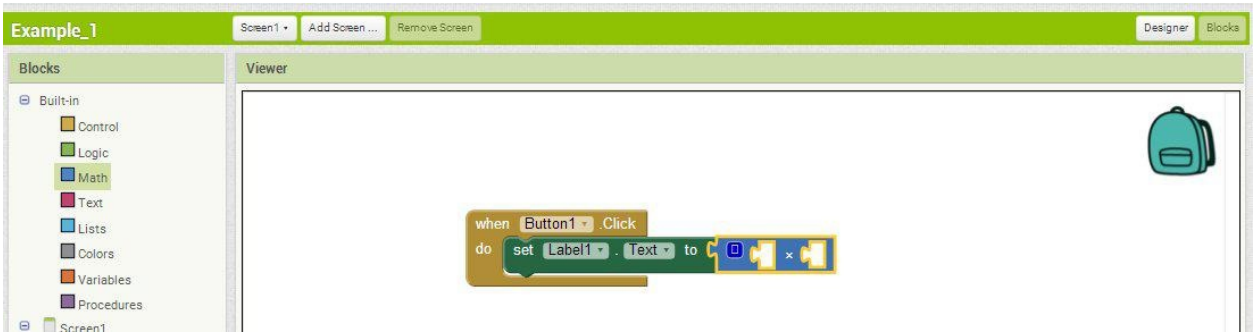
Then Click, Drag and drop 

"**set Label1 . Text to**" to the **Viewer** Window. Positioning it to fit in the "**When Button1 .Click do**" block.

So what have we done now? We've said when the button, "**Button1**", is touched, set the text in **Label1** to read something. What does the yellow around the block mean? Just that, this is the selected block or blocks.

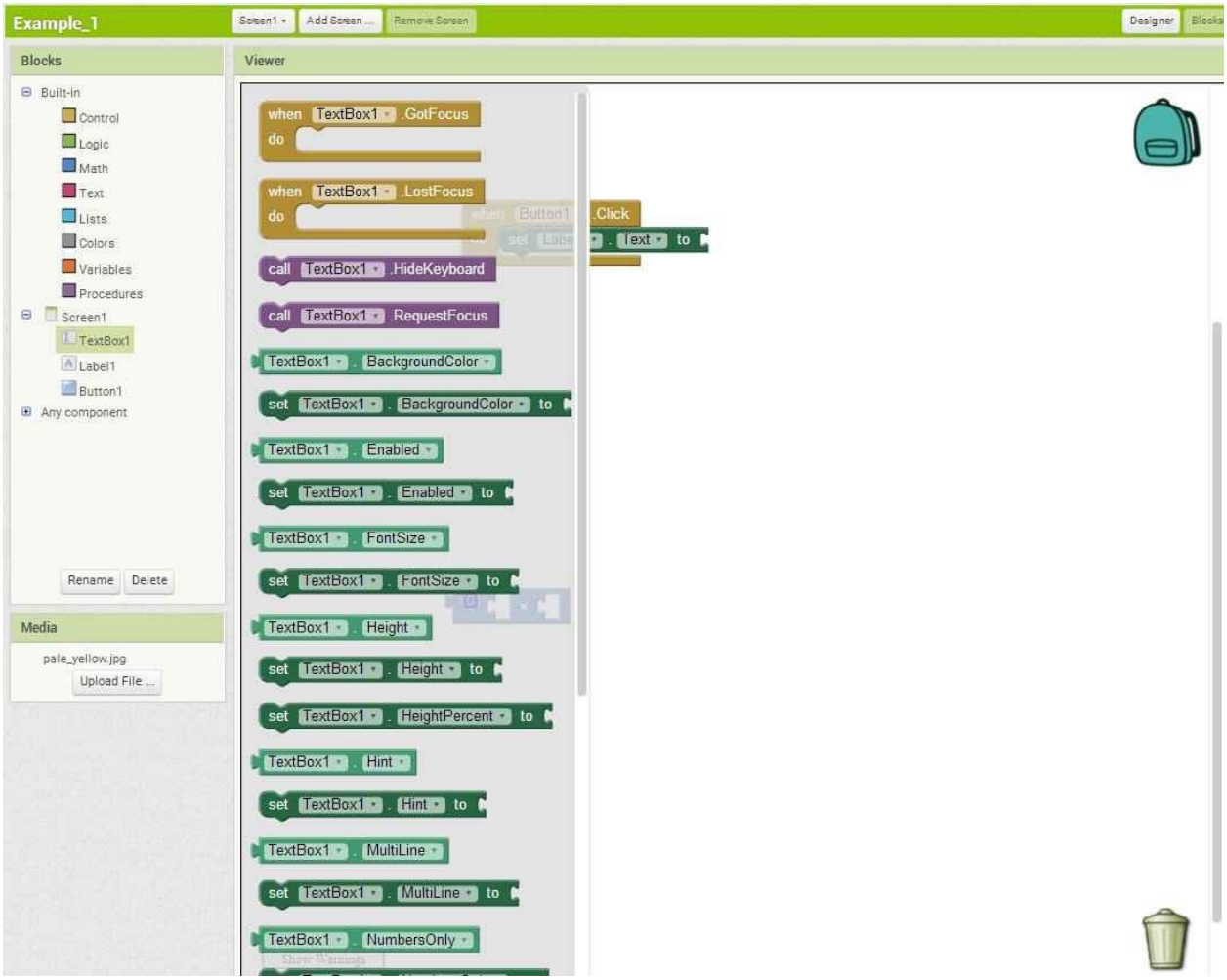
Ok so we seem to be missing something, if we now dragged **TextBox1** in it would make **Label1** display that, but we want it to show that number squared. So we now go to **Math** in the **Blocks** Window in **Built-in** and click that.

Then find  "**multiplication**" and Click, Drag and Drop to the **Viewer** Window. Positioning it to fit in the "**set Label1 . Text to**" block.



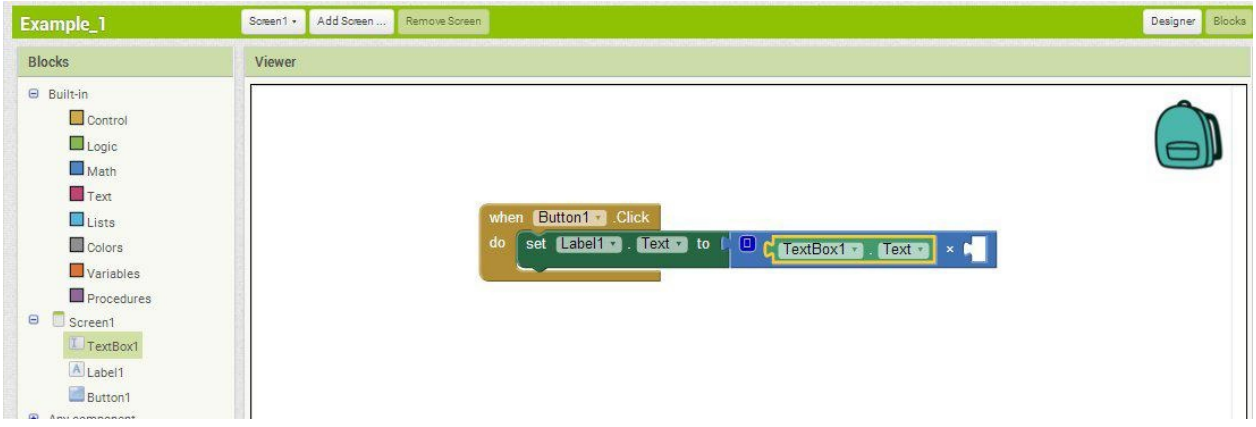
So what have we done now? We've said when the button, "**Button1**", is touched, set the text in **Label1** to read something "**X**" something. So you can probably guess we will go and get the "**TextBox1**" and put it in both holes.

Select **TextBox1** from the **Blocks** Window



Then scroll down until you find 

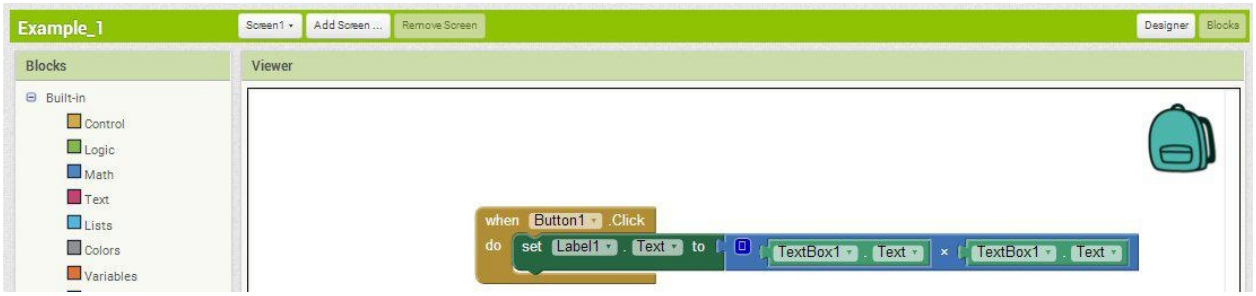
"**TextBox1 . Text**" and Click, Drag and Drop it to the **Viewer** Window. Positioning it to fit in the "**multiplication**" block.
You might have expected to pick "**TextBox1 . NumbersOnly**" but if you do it will not lock in to the "**multiplication**" block, I can't find a reason for that.



One more thing to do here either repeat the last stage or right click you mouse on 

"**TextBox1 . Text**" and select "**Duplicate**" and then drag the duplicate 

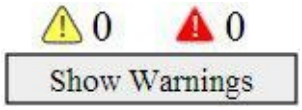
"**TextBox1 . Text**" into the last empty hole.



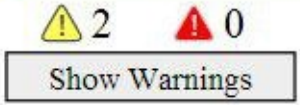
Select **Projects**, in "**Command Bar**", and then **Save Project**. To be safe, although it seems to save as your going along anyway, but if you want to be safe **Save Project**.

TextBox1 . Text So what have we done now? We've said when the button, **Button1**, is touched, set the text in **Label1** to read the solution to **TextBox1** multiplied by **TextBox1**.
Or as it says on the pieces themselves "**when Button1 click do, set label1.text to textBox1.text times textBox1.text**" so they've made fairly straight forward to construct commands or procedures.
There is another way of doing this last stage which we'll cover later in this section.

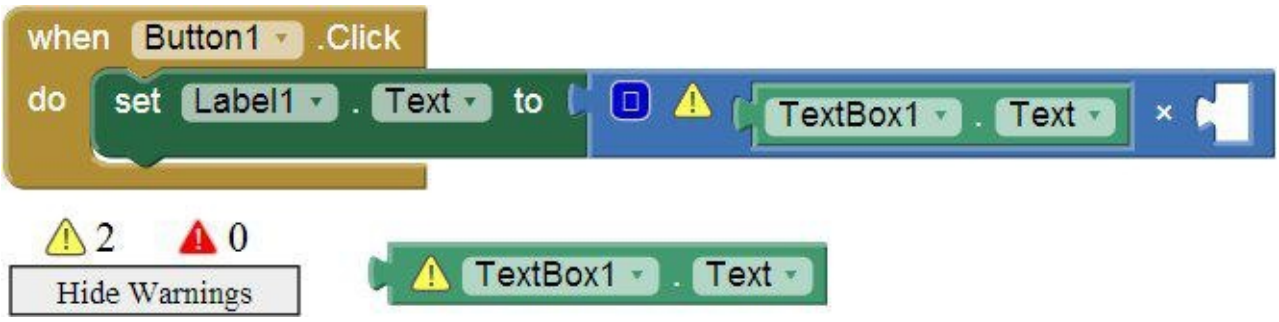
You'll notice in the "**Viewer**" a few other Icons:



This is the system check, before you think you've finished this will tell you it's status, the "**Red Warning Triangle**"  are serious and your App will not work. The "**Yellow Warning Triangle**"  are warnings of non-catastrophic errors, for instance.

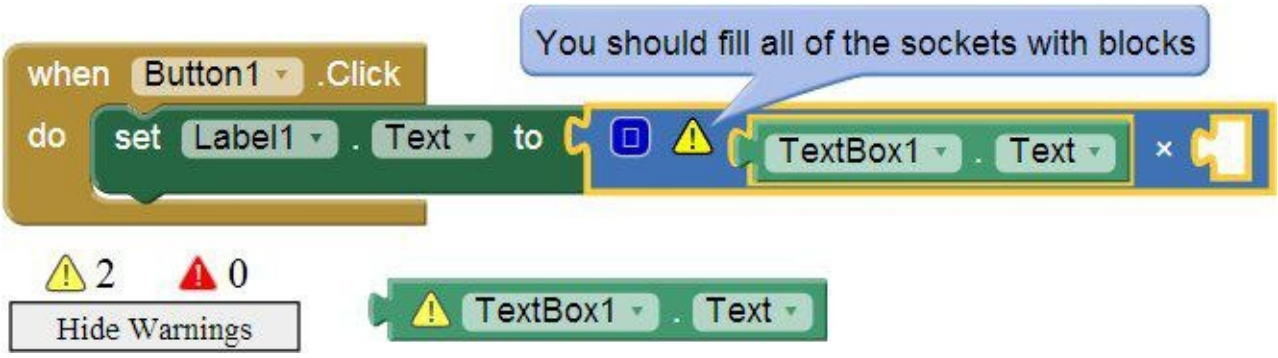


This is telling us there are 2 errors, they are not serious, but in this case the App will not work out the correct solution.

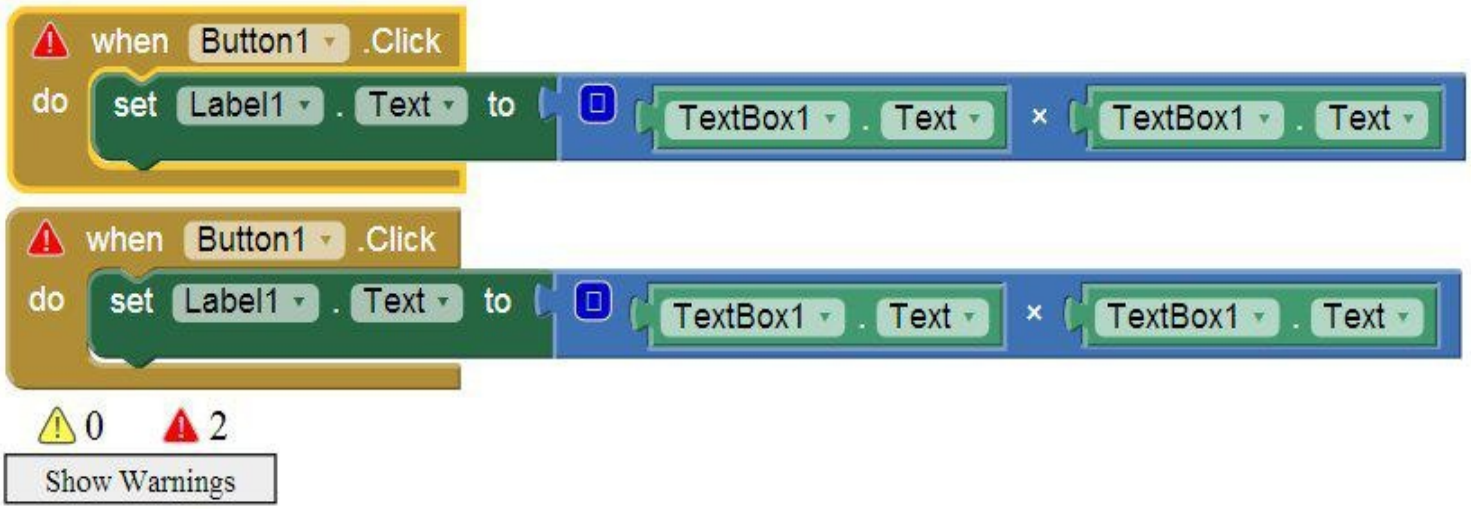


If we click on the "**Show Warnings**" we get this, it highlights any Warnings. It's obvious in this example but if there is a lot of blocks this might be a help. Be aware this doesn't mean the App won't work.

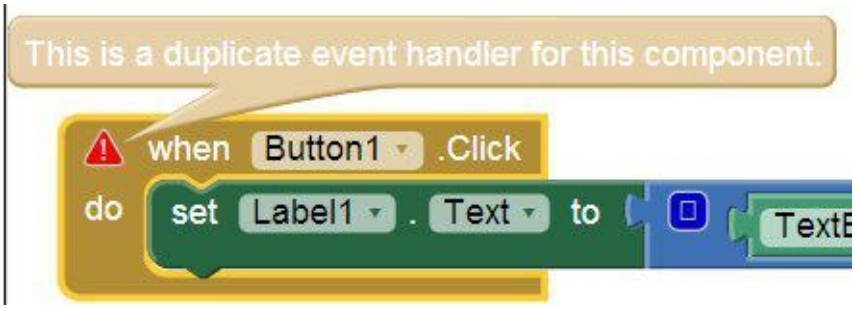
If we click on the "**Yellow Warning Triangle**"



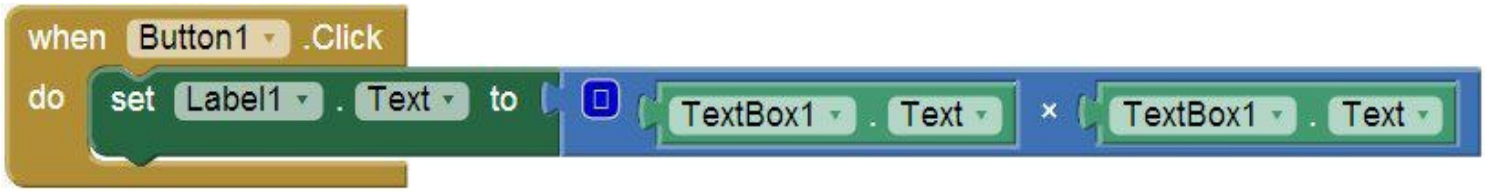
We get this, a warning Message, telling us we need to fill all the sockets with blocks.

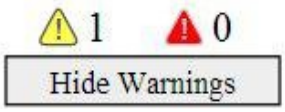


If we click on a "Red Warning Triangle"



We get a message telling us what the warning is, in this case there are duplicate event handlers for Button1. This means it won't work as every time you touch Button1 it would have to try 2 do things at the same time.



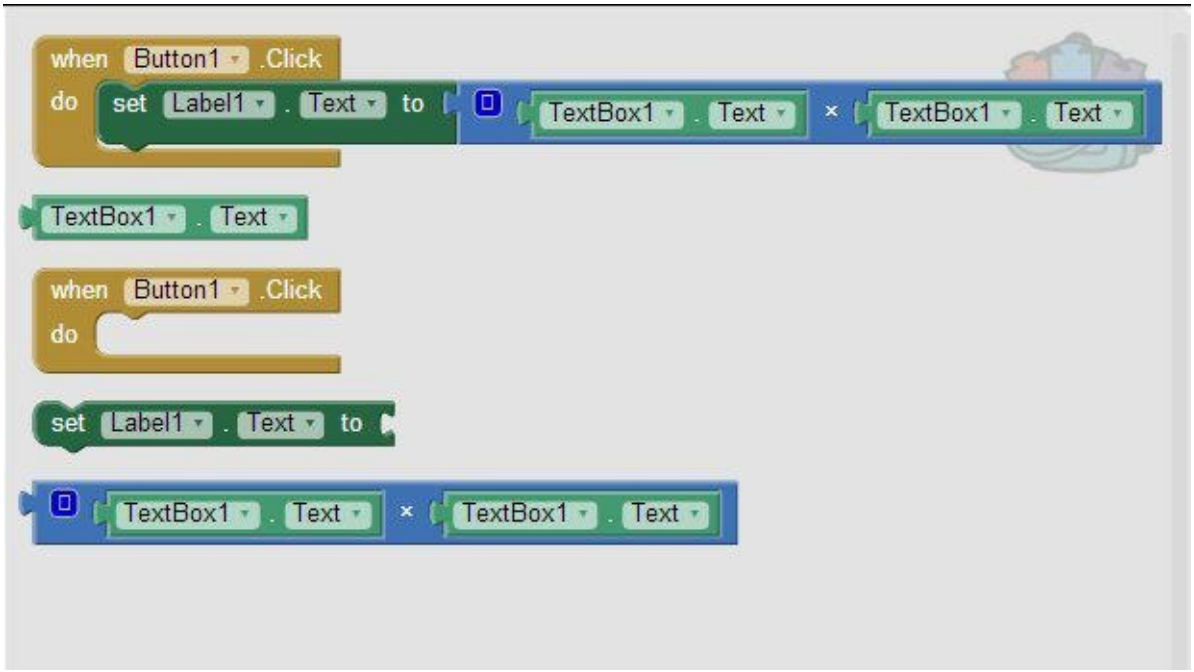


Here I've left an extra block in case I need it later, this will not impact on the App working but may increase it's size slightly. Why would we do this? When you're creating Apps with a lot of **Blocks** it was common to leave odd ones about, that were going to be used a lot, to save going to find them.
Which bring us on to the BackPack or Temporary Store.

If you drag and drop a component, or an assembled block of components, to the Icon, the Icon will change to this, to show something is stored. The component or components will still also be in the **Viewer** window.

This is only temporary, once you log out, they are gone and next session you have load them up again. But this will not cause any Warnings as they are not in the App and will not use any space

because again they are not in the App. If you click the Icon it will open to show you what it has stored. Click what you want and drag it into position, you will notice it's still in the store so you can use it as many times as you like from the store and it will still be there, but only during that session. You can put blocks in from one project and get them out in another project but unless the same names are used the blocks may not work and might need replacing or renaming.

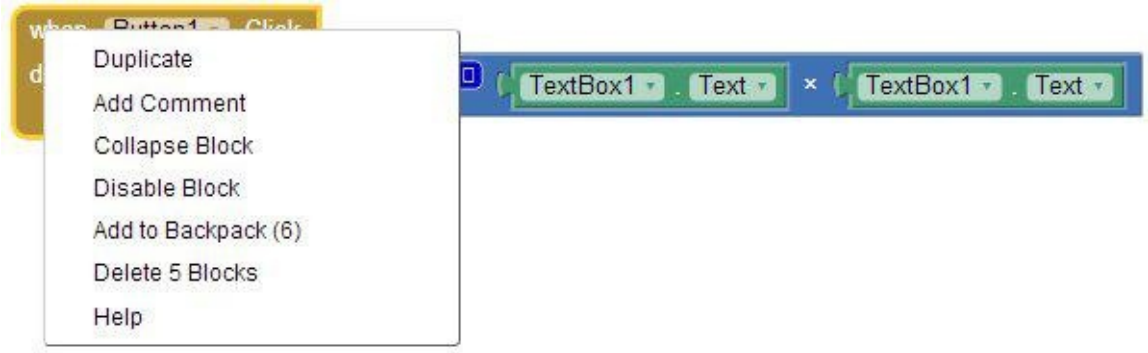


Finally we come to the Bin or Trash Can, it's almost too obvious but I will say it, if you want to get rid of something drag it and drop it in the bin and it's gone, only from here it still exist in **Screen1**.

If you drag more than one block i.e. **Math** Multiplication block and the 2 **TextBox** blocks you will get a "Are you sure you want to delete all 3 of these blocks?". Click "**OK**" and they are gone.

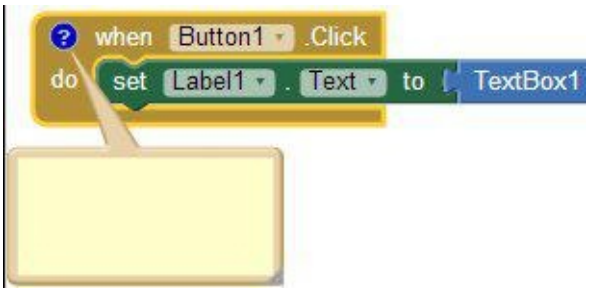
With a single block, drop it in and it's just gone, no questions asked.

Before we go I guess it's worth mentioning "**Right Click**" - If you Right Click on a Component you will get the following Options:

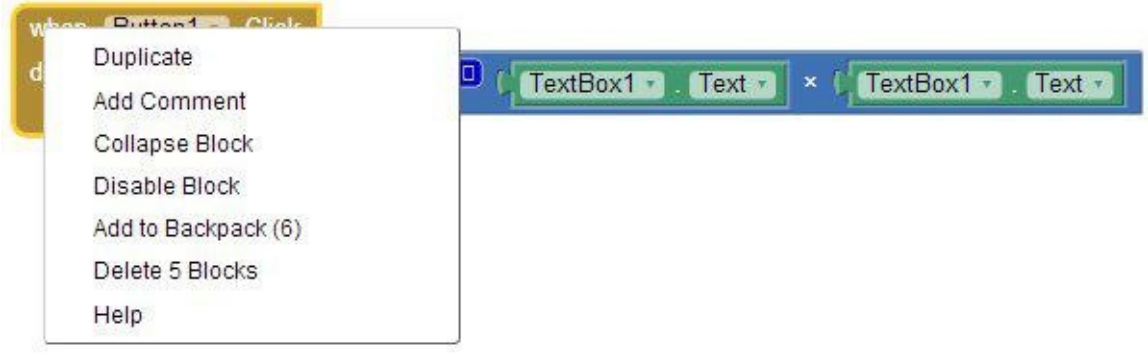


If you right click on the **Button**(or another) Block and choose "**Add Comment**"

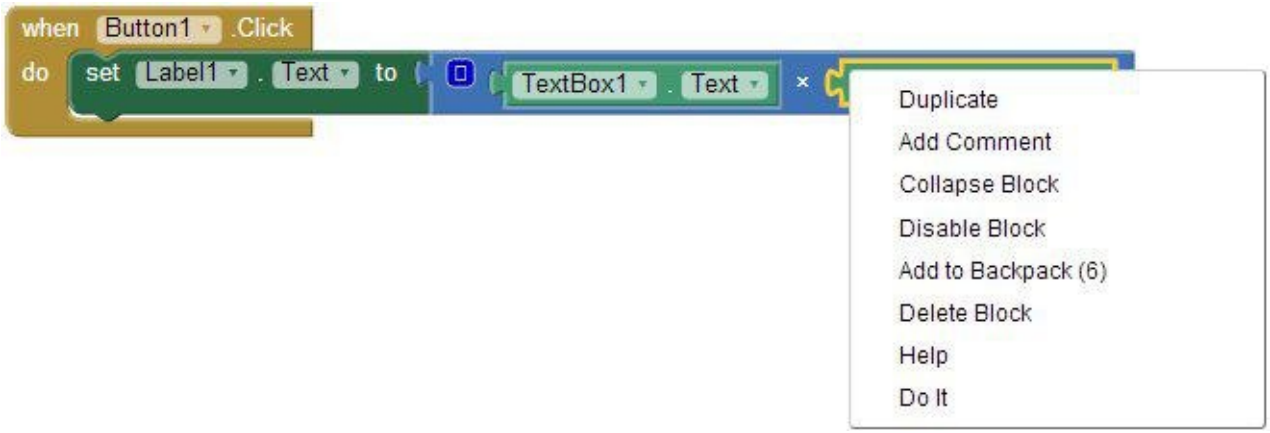
As you see above, a  Question Mark appears.



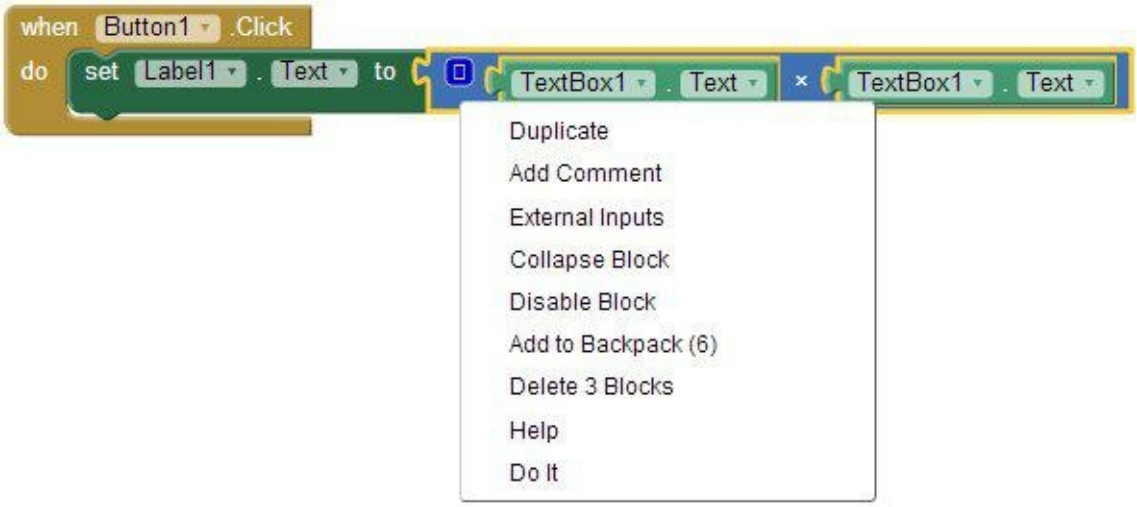
Click the  Question Mark and a box will appear that you can click into and add text. You can leave yourself or others build notes. Be aware that this may increase the size of the App even if it's by a very small amount.



Above I clicked on the **Button** block and I effectively have the complete command, so if I delete, I delete all five blocks.



Above I clicked on the **TextBox** block and I have only the one **TextBox** so if I delete, I delete only the one **TextBox** block.



Above I clicked on the **Math** Multiplication block and I have the **Math** block and the 2 **TextBox** block if I delete it will delete 3 blocks the **Math** Multiplication block and the 2 **TextBox** blocks.

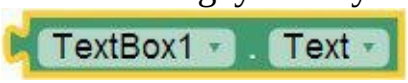
Just quickly running through this if you duplicate there will be 2 of what ever you click on. "**Collapse**" will collapse the block to the point you selected i.e.

Selecting the **Button** looks like this with nothing else showing. This is great, you can use this when you finish an assembly of blocks to keep them out of the way and not accidentally pick something you already finished.

Selecting the **Label** looks like this, we see the **Button** unaffected with the **Label** but nothing else showing. And so on.

"**Delete**" will delete whatever you've selected, it always tells you how many blocks you have selected, so be warned.

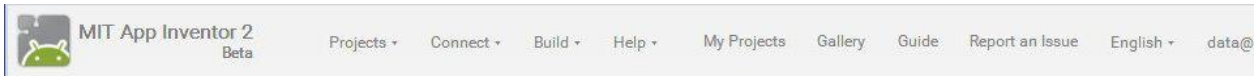
"**Add to Backpack**" - You can add everything you pick to the Backpack, or disable it. You'll note in the previous right click examples it says "**Add to Backpack (6)**" so you might think there are 6 blocks already in there. There is actually 6 items so that could be blocks or assemblies of blocks or a bit of both.
An important thing to remember is that you can put blocks in the Backpack in one Project and then go into another Project and get them back out. But, and this needs to be remembered the blocks you pull out my not exist in the new project so you will have to go through and either replace or recreate them. The main use for this is to pull in things you use in a few places like checking routines and calculations or just ripping out something similar you did in anther App.

Another thing you may have notice is the little pull-down buttons(tri-angle pointing down) 

in a few places on the blocks, this just allows you to change between TextBox1, Label1 or Button1 and any other of those you have created. So if we had two Buttons, labels and TextBoxes we could Duplicate the whole Button1 Block Assembly and then go through so we have Button2, Label2 and TextBox2 × TextBox2 or whatever we wanted. You will see one on the "**.Click**" and on the "**.text**" parts of which give you the other options you had when picking from the whole Block menu.

First Test

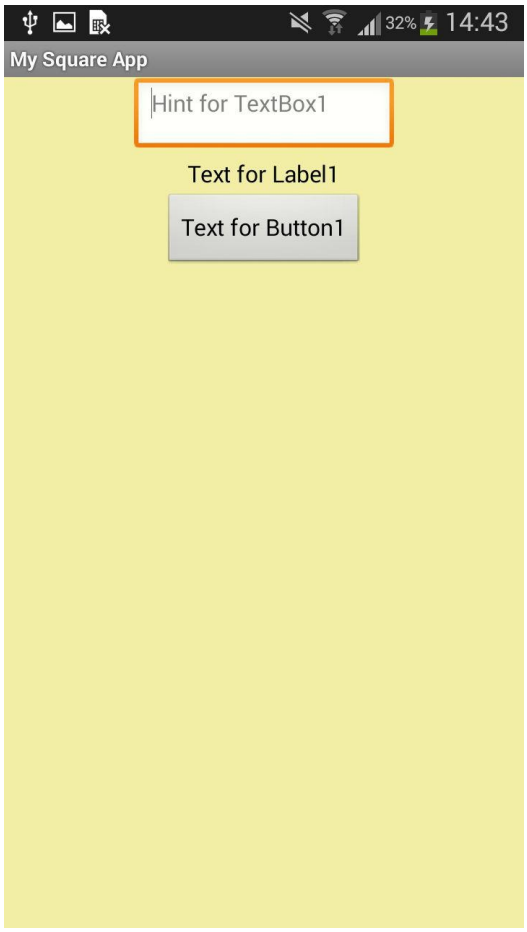
Go to the Command bar



And Select "**Connect**".

You now have the choice, the on screen Emulator or a Device either connected through Wi-Fi or Cable/USB.

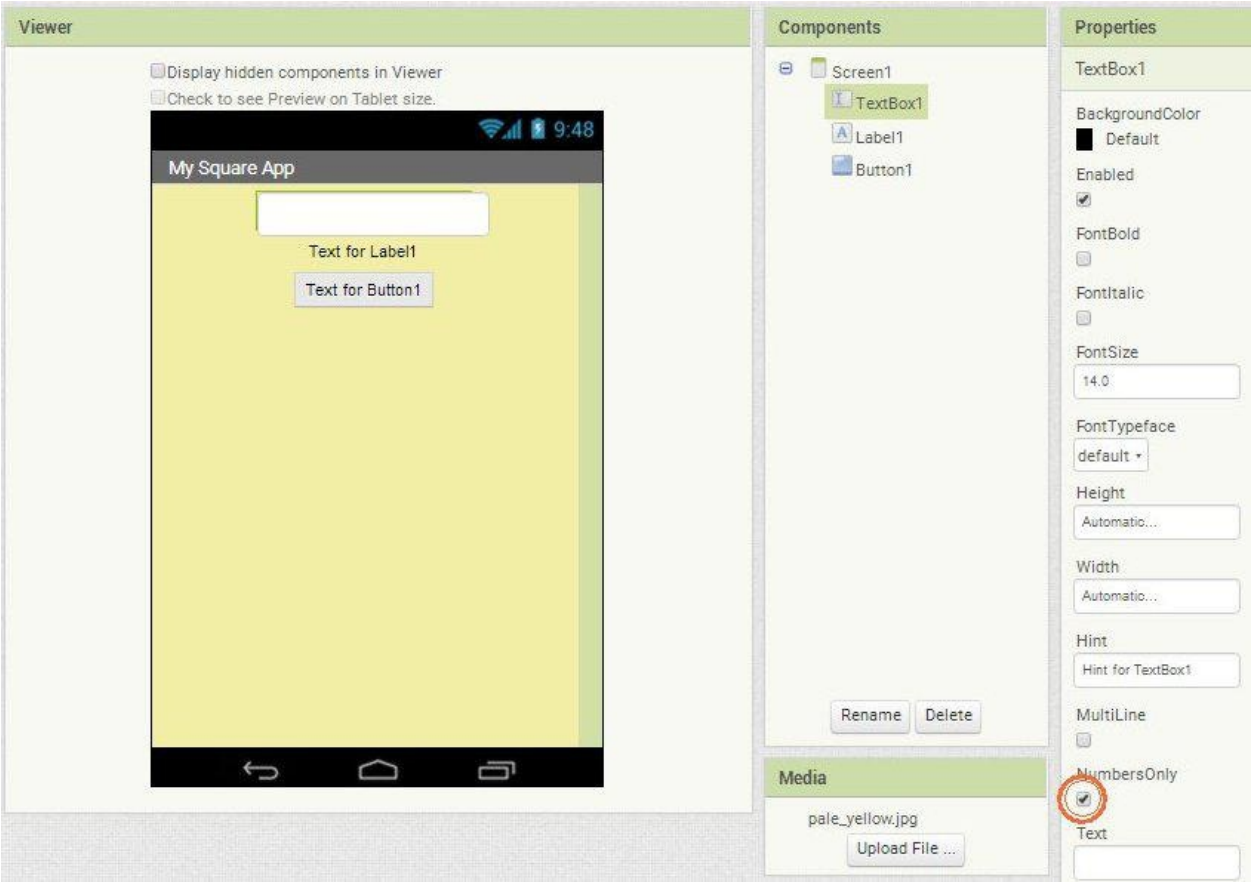
See Installing the App and Using the App



It will look like this on a phone and very similar on the Emulator. Now touch the Input box, where it says "Hint for TextBox1".

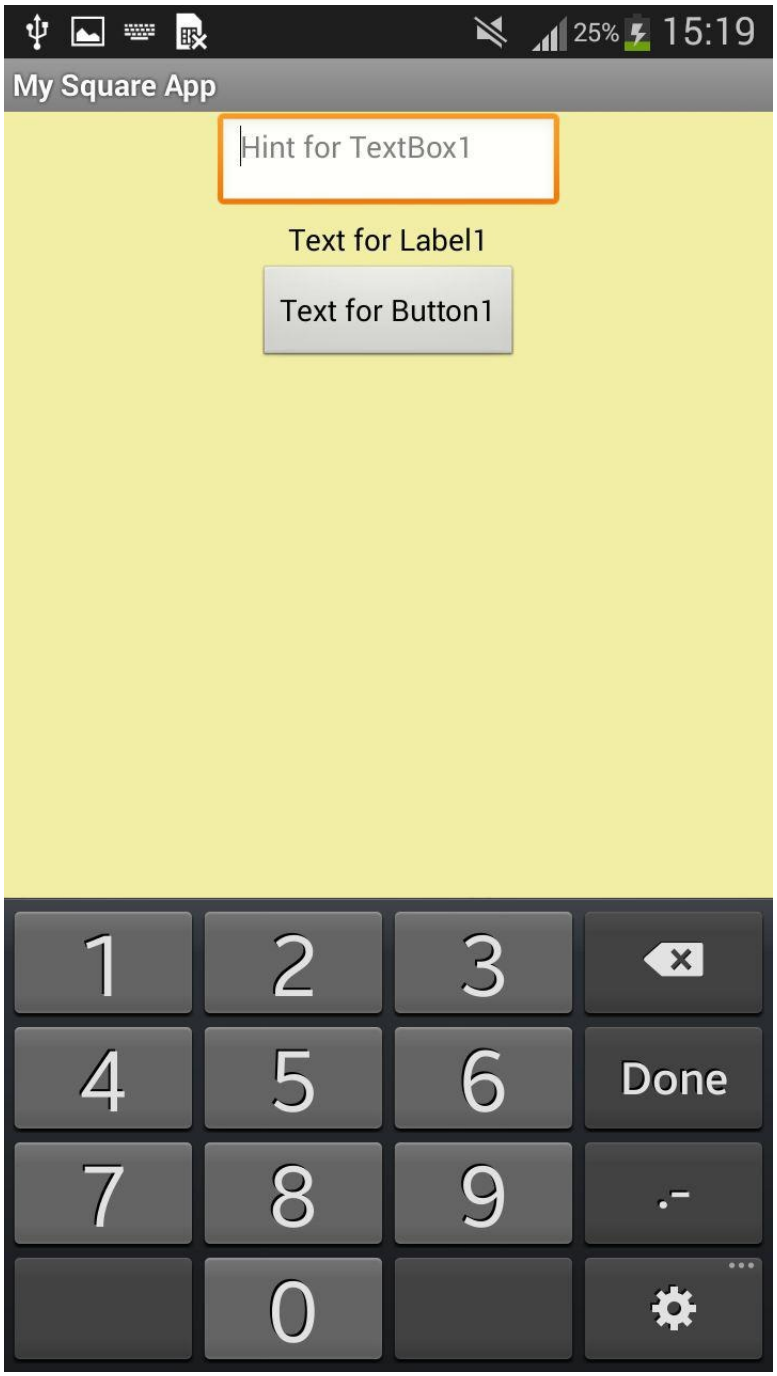
It will look like this on the Emulator and again Similar on the Device.

The first thing you'll notice is we have a QWERTY keyboard come up for entering our number. If we leave it like this then it's too easy for you or your user to crash the App by putting a letter as a number. Plus every time you use it the first thing you'd have to do is select the numeric or hold the top row of key until the number comes up, not good. So if you now, back on the computer, click on "**TextBox1**" in either **Viewer** or **Components** and it's properties will appear in the "**Properties**" Window.



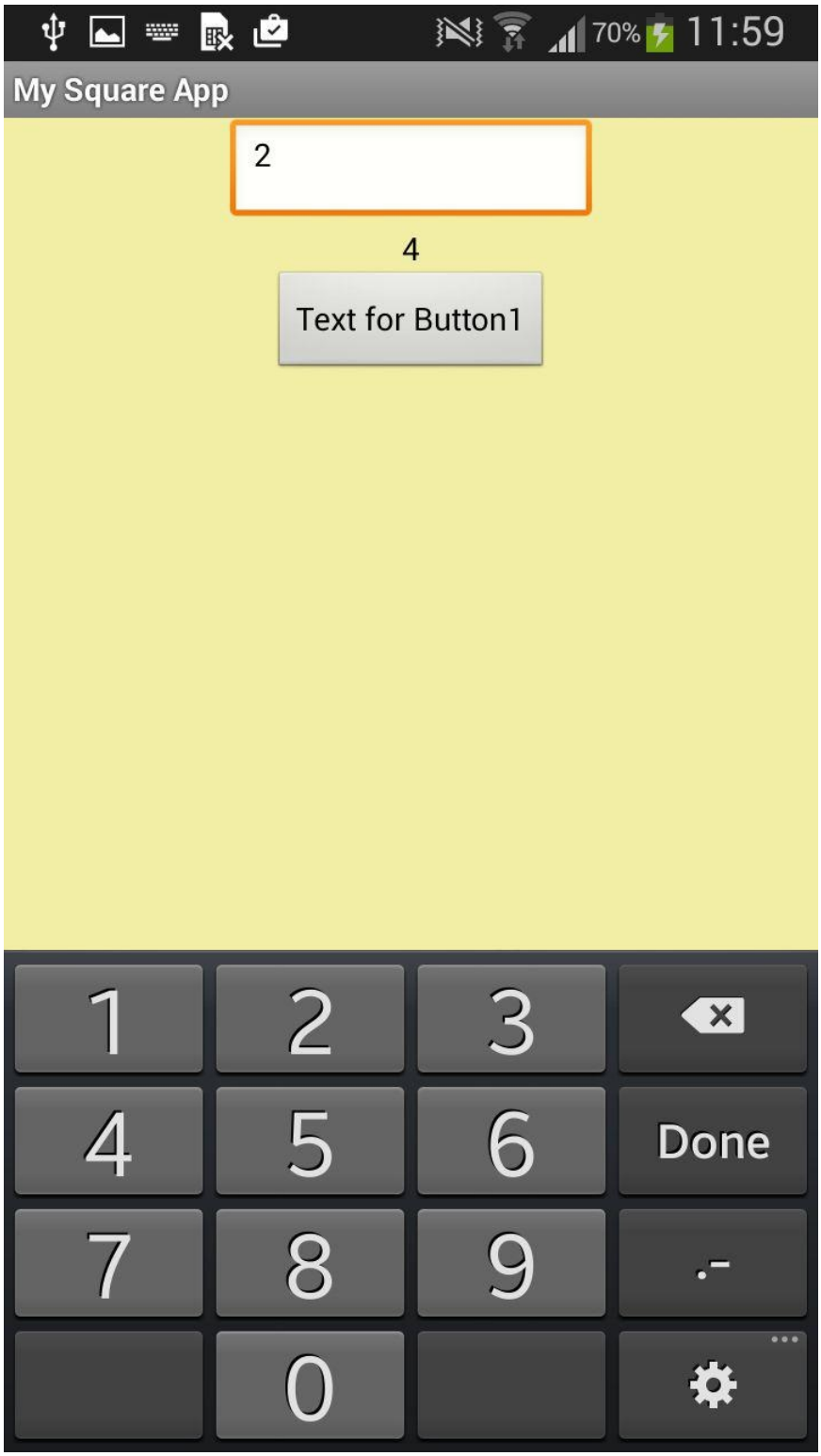
Look down to where it says "**NumbersOnly**" and click the box.

Above is how it now looks on the Emulator.
Even if you plug a keyboard in, your Device should now only accept numbers.



And above is on a phone.
You will notice that the Emulator isn't the same as the phone and so I will be using the phone images as this is what it will look like on most Devices. If you can't connect your device don't worry the Emulator is good enough, but it won't give you the true look so you'll need to check by downloading a copy of your App on to the device you want to use it on, every so often.

I guess before we do anything else we should check that the function works correctly. So on your Device touch "**2**" and then either "**Done**" and "**Text for Button1**" or just "**Text for Button1**"



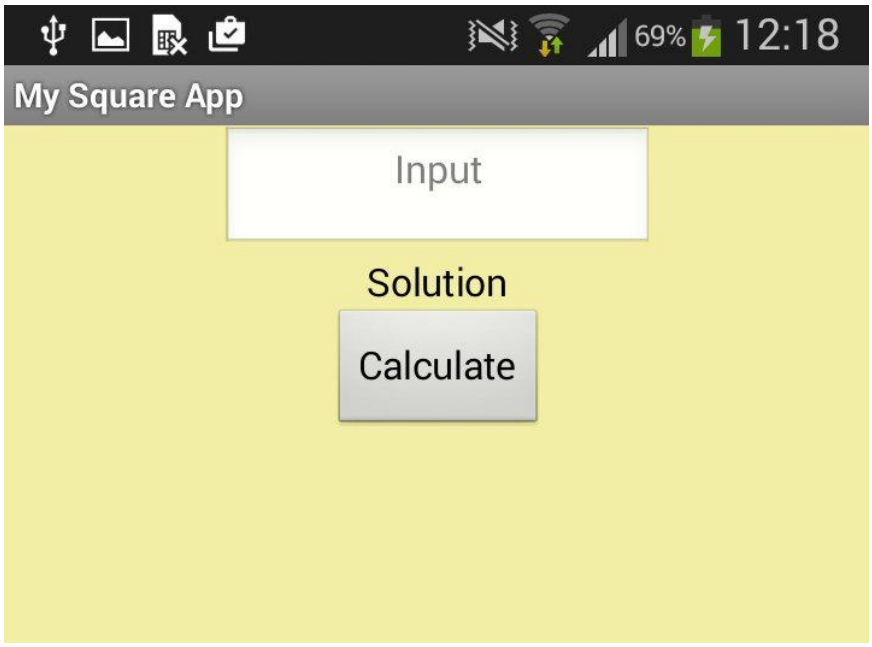
So it works, it still doesn't look very good but it does work.
If you touch "**Text for Button1**" without putting a number in first you'll get this

We can create a line to stop this later, you'll also notice it still has the previous answer, so we may want to add something to clear that at the same time.

Now if you look back at the Designer/Viewer screen we can start changing the appearance to start with "**Hint**" you will see "Hint for TextBox1" we don't want that, so we either delete it or replace it with something like "Input" or "Number"

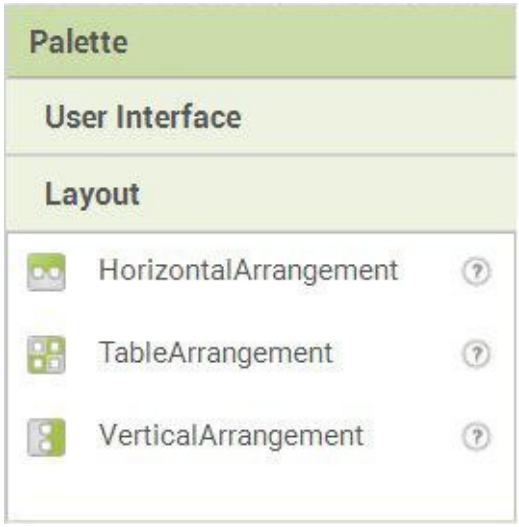
We could "center" it using The "TextAlignment" option and do something similar with the other

Components.

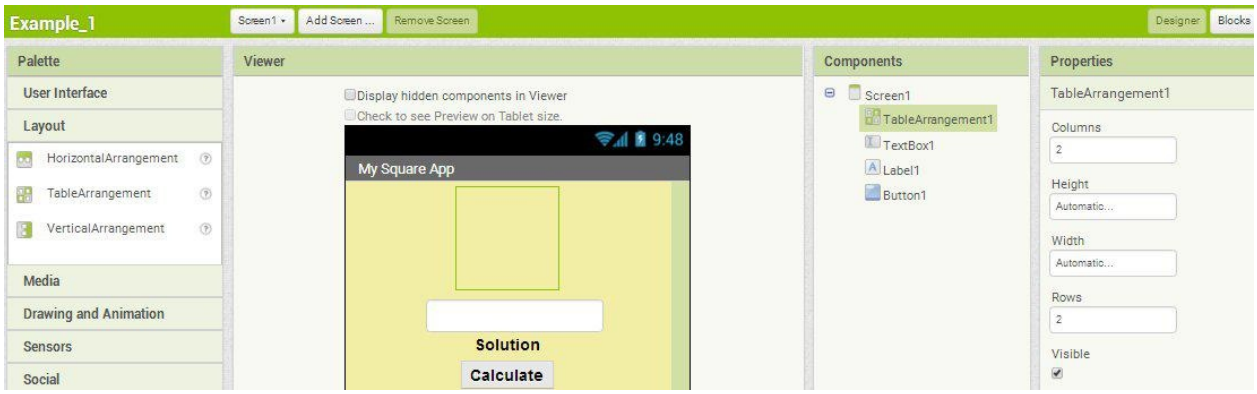


Still not a great look, but getting better. The way I do it is to use a Table type Layout and maybe change the order and add more labels.

So we will use **Layout** from **Palette**



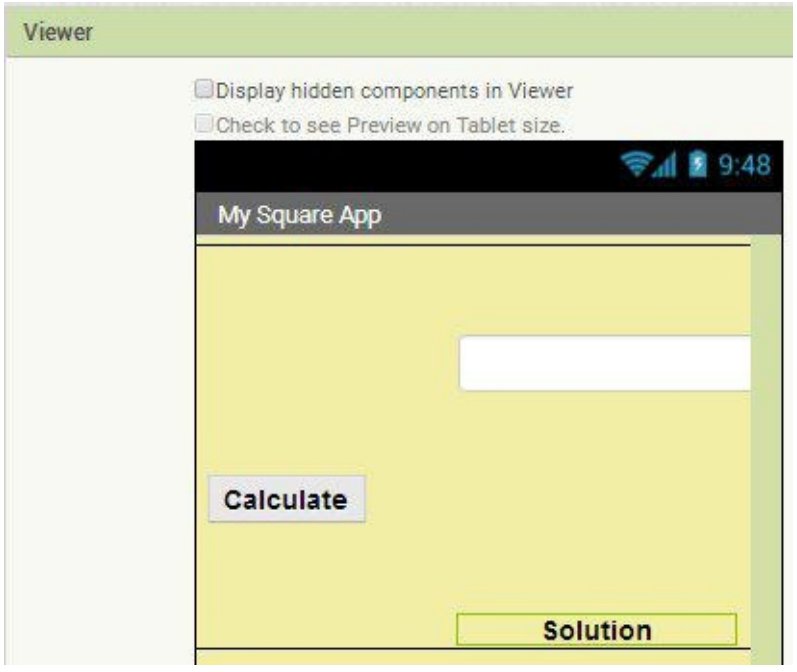
Why you may ask? Well we can use **TableArrangement** which will give us a table like structure to place our **Label**s,**TextBox** and **Button** in if we want and I do.
Select the **TableArrangement** Icon and drag it to your **Viewer** above "**TextBox1**"



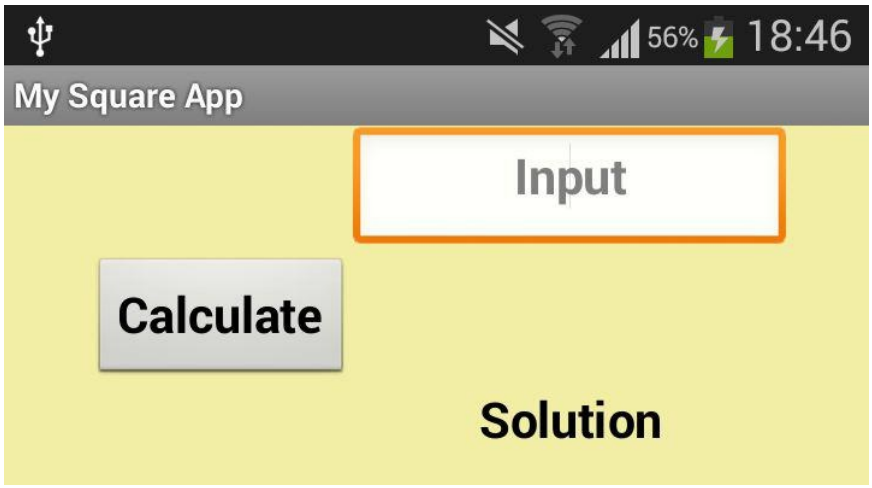
We now have **TableArrangement1** in the **Components** and a new **Properties** list.

Lets change the Columns to 5 and rows to 6, this gives us room to mess about with. We can change these at a later date if we don't like what we get.

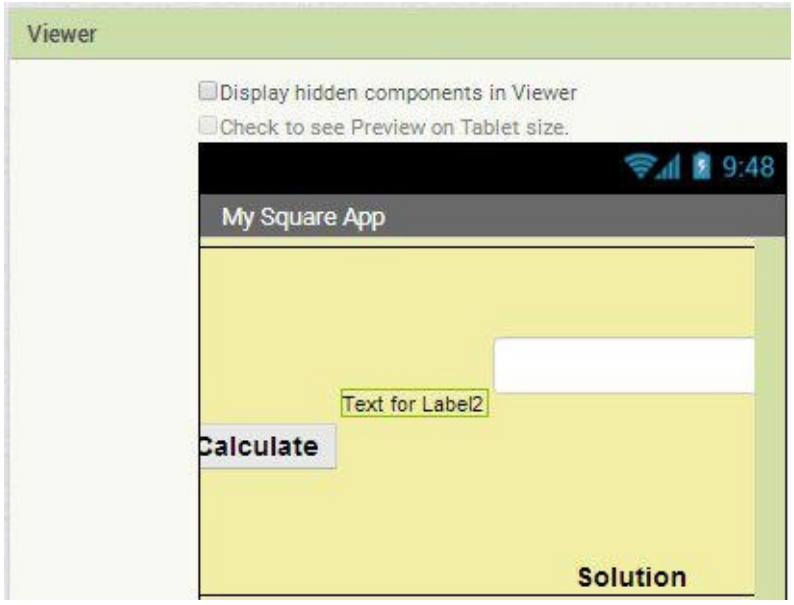
Now we need drag **TextBox1** to the 2nd row down and 4th from the left. Drag **Button1** to the 4th row down and 2nd from the left. And drag **Label1** to the bottom row and 4th from the left. It will look some thing like this.



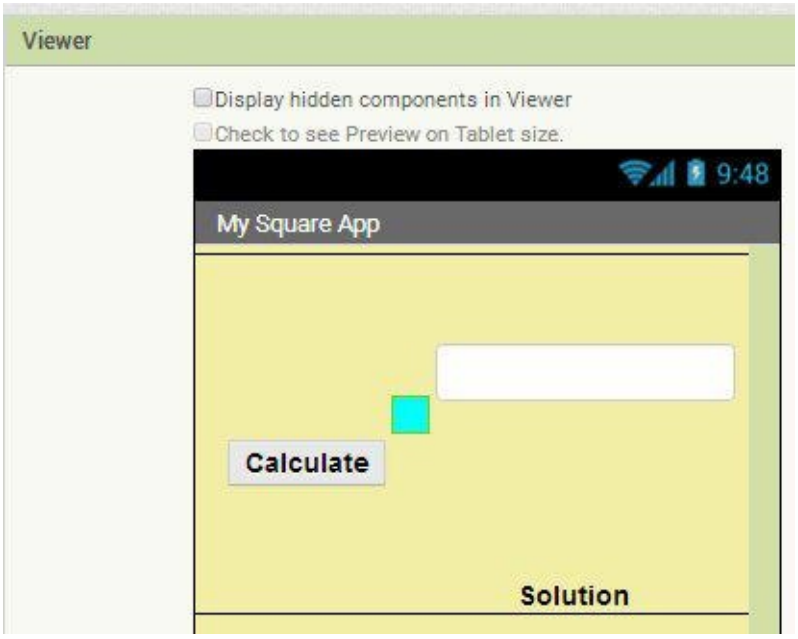
On the Device it will look some thing like this.



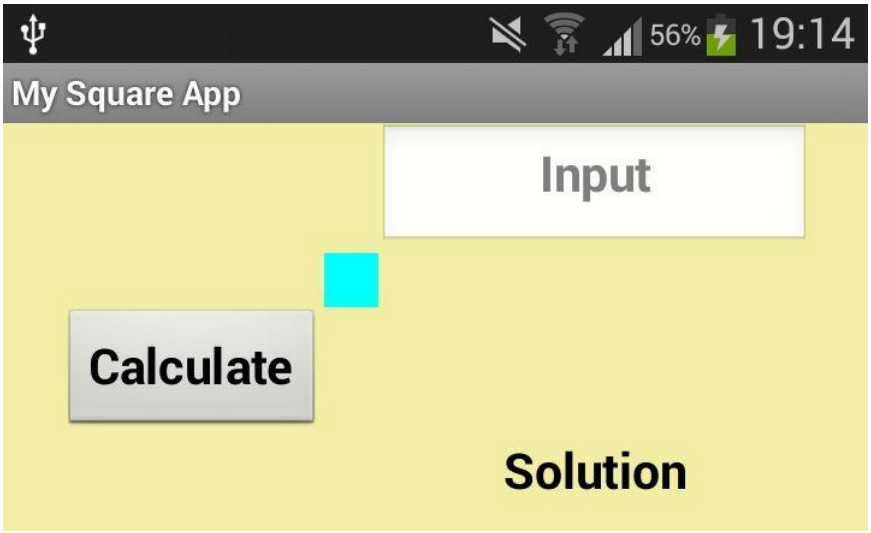
This because it's table cell format is flexible so if we want things spaced out we have to add more **Label**s maybe with no text but with a set height and or width.
So lets drag a new **Label** from the **User Interface** the **Viewer** and place it 3 rows down and 3 columns from the left, right between **TextBox1** and **Button1**.



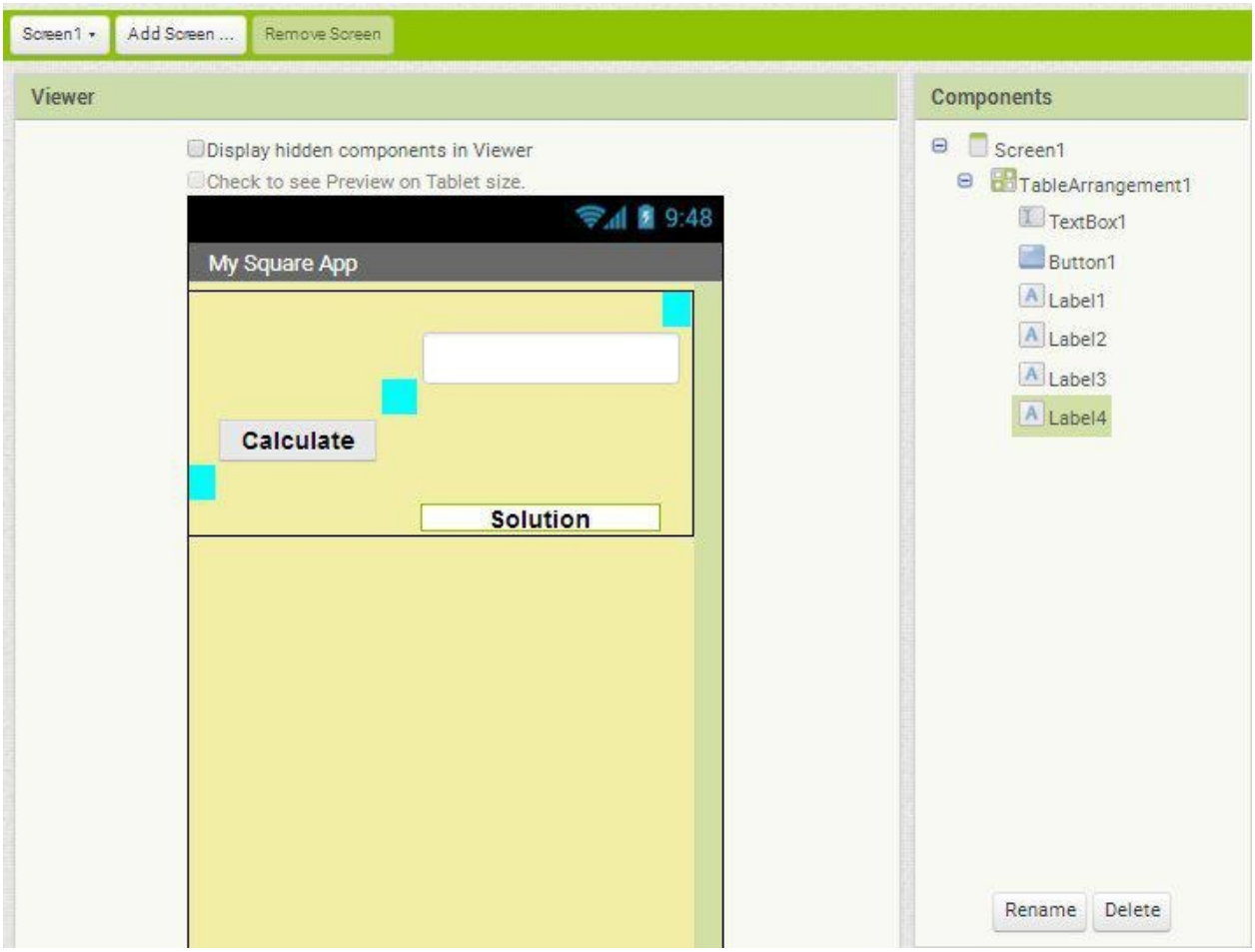
Now select **Label2** and in **Properties** put in 20 pixels for height and width, and delete "Text for Label2" from the **Text** box. I've given this a Cyan **Backgroundcolor** so you can see it.



Straight away you'll see we can space things out.

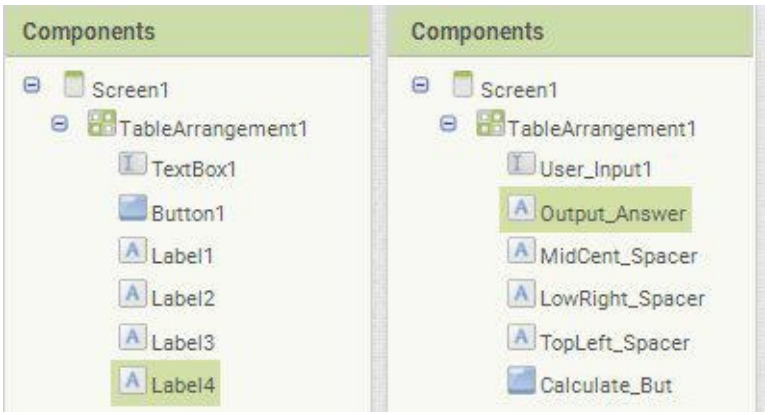


Above is what it now looks like on the Device. It's still not what I want so I will add another **Label** above **TextBox1** to space that out this time I will only give a Height. I've also gone in to **Label1** and made **Backgroundcolor** White.



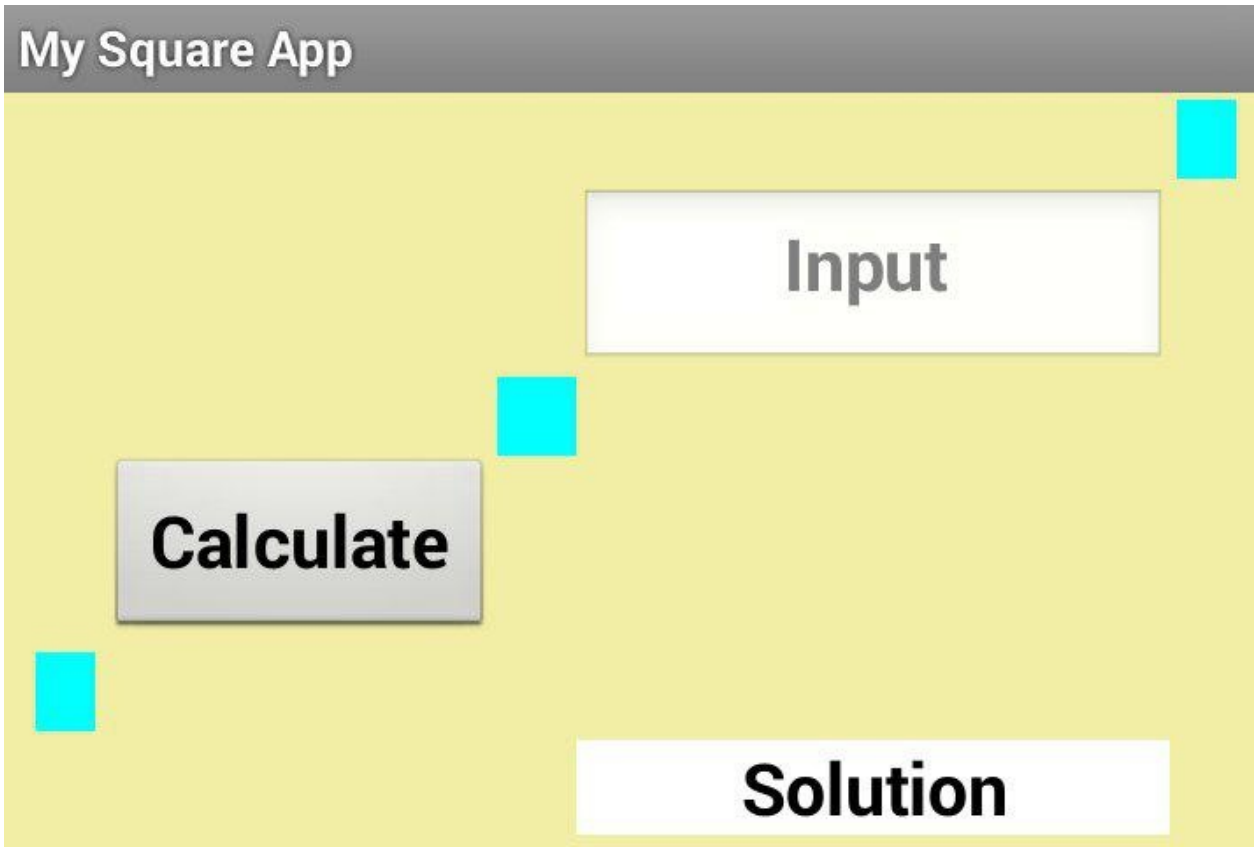
This is probably a good time to mention Component Names when you look down the **Components** list you'll see **TextBox1**, **Button1**,**Label1**,**Label2**, **Label3** and **Label4**. If we carry on like this we will end up with tens of them and we'll struggle to remember what they are. So select **Label4** and then Click on the **Rename** Button below it.

Overwrite **Label4** and call it something you'll find easy to recognise later.
Like **LowRight_Spacer** and click **OK**. Now do the same with Rest.

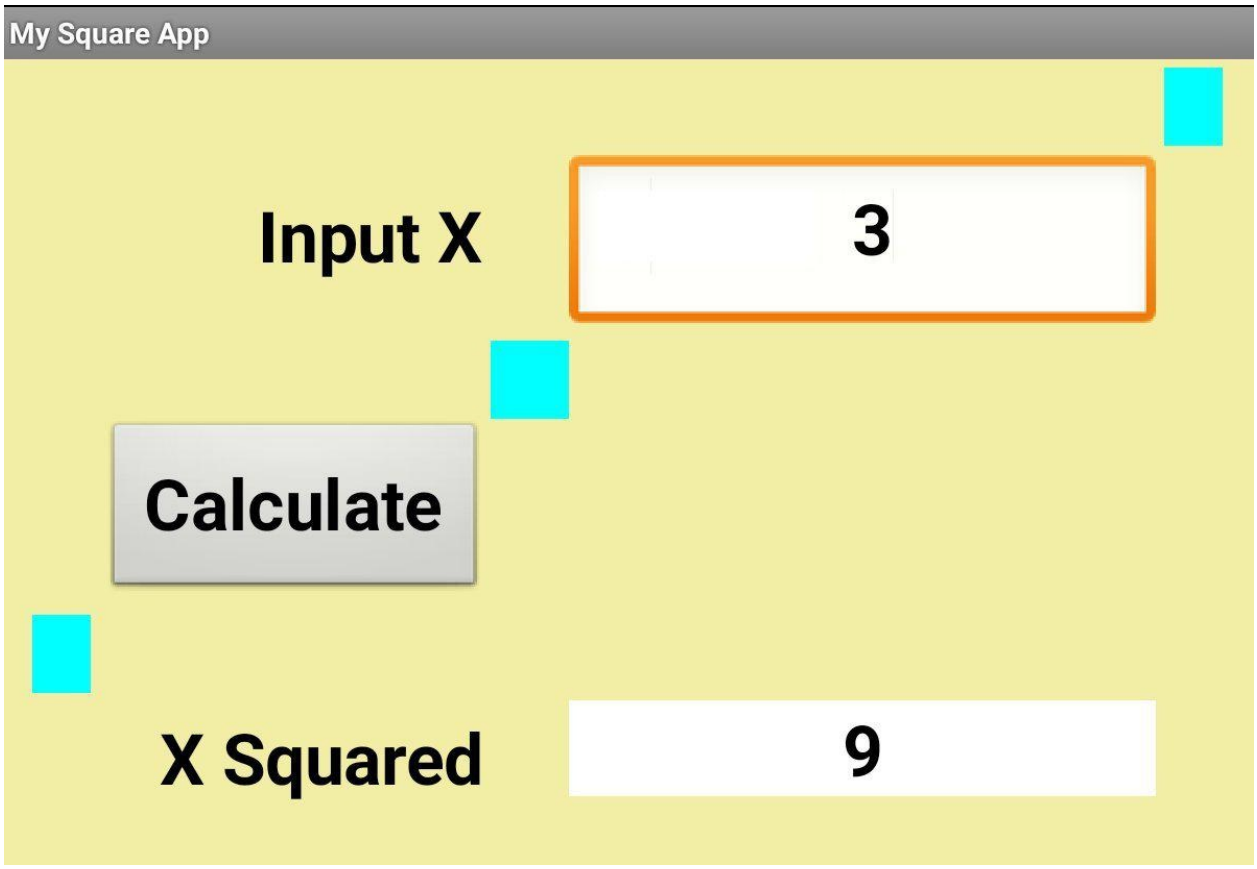


On the left is the before and on the right the after. These should be easier to follow when we have more Inputs and Labels. But you might ask what about our "Blocks"

When you change from the **Designer** screen to the **Blocks** screen, you'll see that all the names have updated automatically in the Blocks and Viewer windows. You may also notice at the bottom of the Blocks window, that you can also Rename or Delete blocks from there.

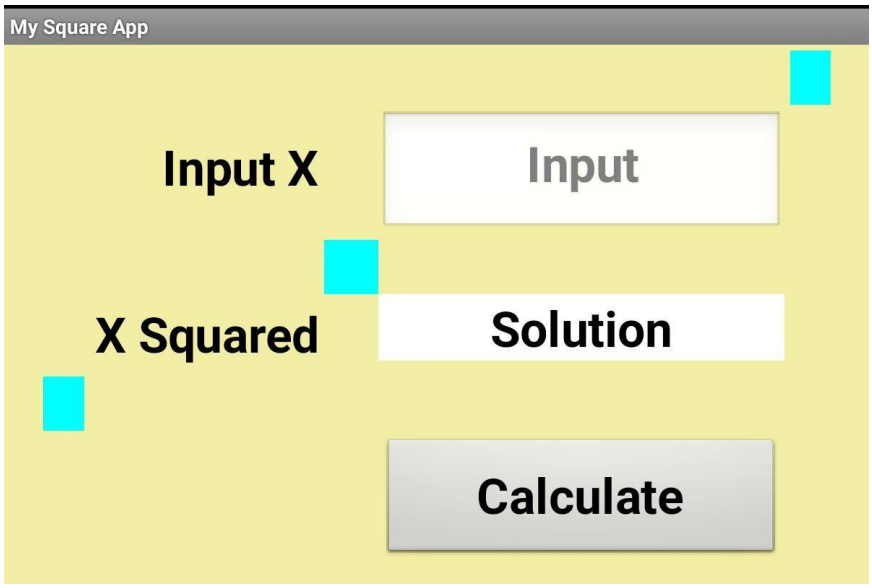


Back to **Designer** screen, now we might want to add some general text. So drag 2 more **TextBox**es over, one, 2 to the left of Input and the other, 2 to the left of Solution. Change the Text to sometime Like "Input Number","Number to Be Squared", "Input X" or whatever you want for the top one and "Answer", "Solution" or "X squared" for the lower one.
I've changed all of their **FontSize** to 18 and ticked **FontBold**.

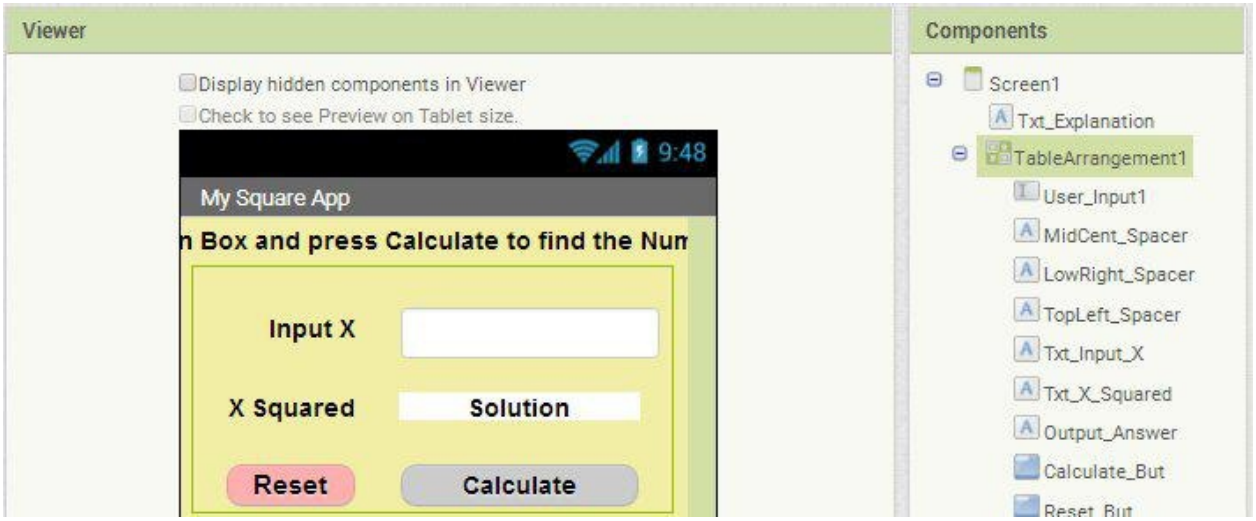


Getting Better, but still not right so I'll pull down the **Calculate_But** to where "X Squared" is and move that and Output_Answer up on the middle row.

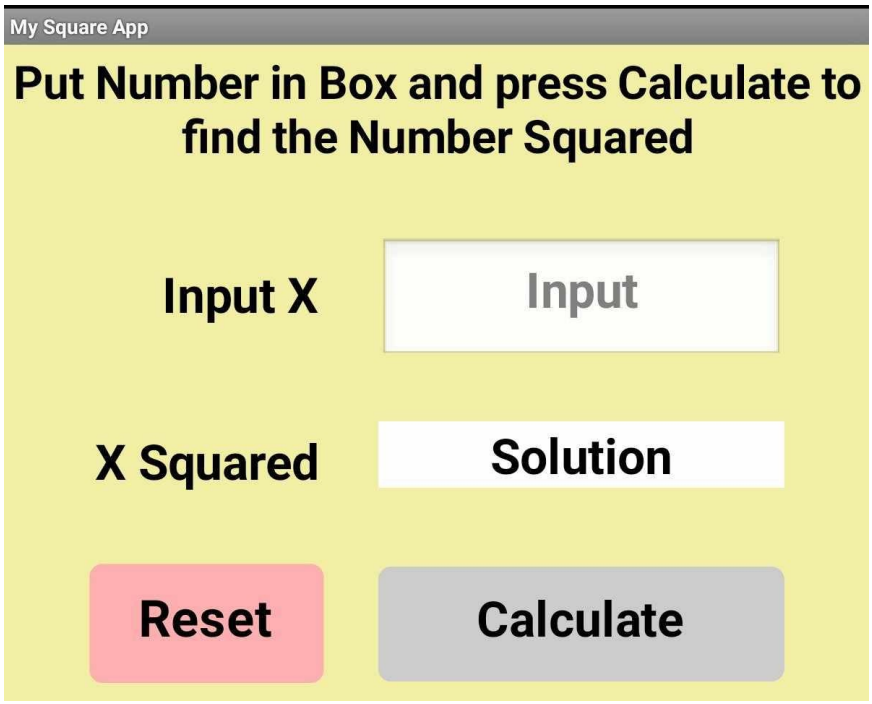
You'll note we have 2 new **Labels** so they will need names.



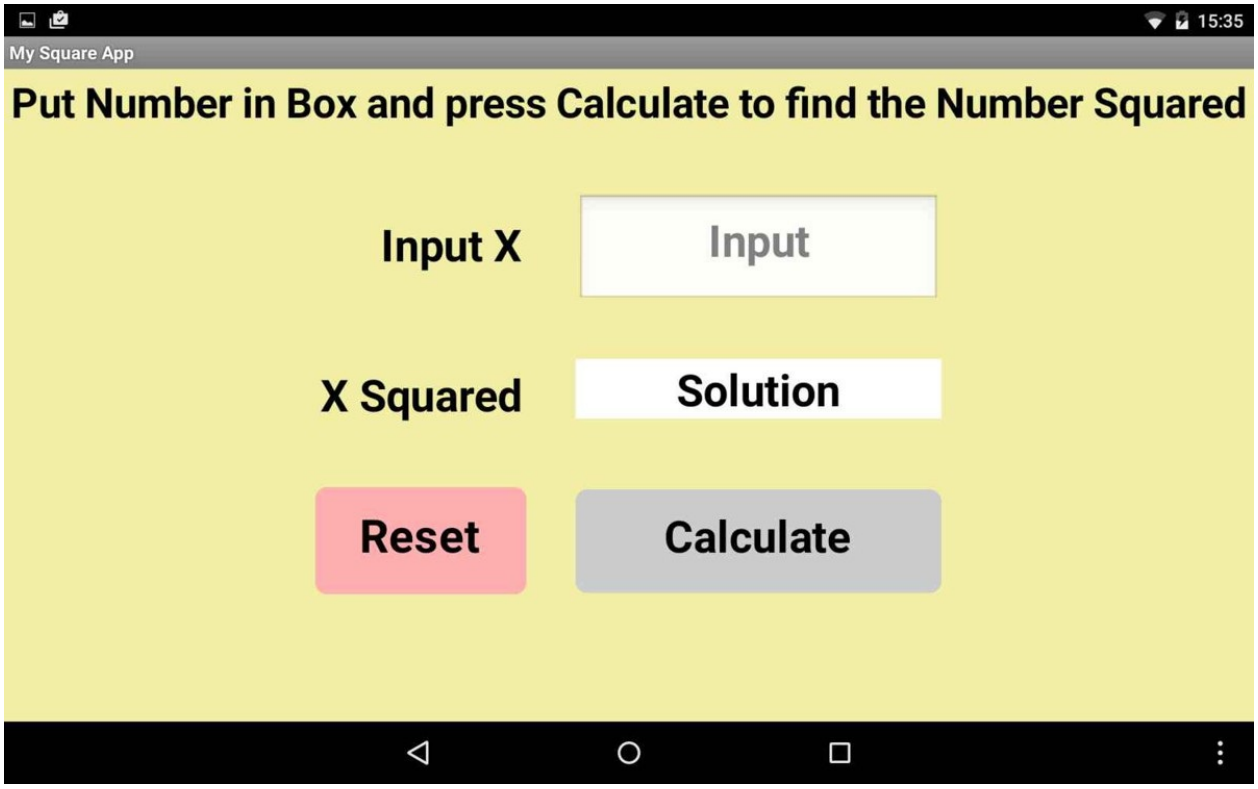
I think that's a bit better.



I've now changed the spacer **Backgroundcolor** colours to None and added a **Label** above the **TableArrangement1** To give an Explanation of how the App works and Renamed it. I Also Added another Button, **Reset_But** which we can program to reset the input and output. I also changed **Backgroundcolor** for **Reset_But** and **Calculate_But** and Change their **Shape** to "Rounded".

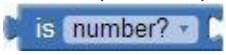


That's Looks better. Above is in Portrait and Below is Landscape. The text at the top look ok on the Device but goes off screen on in the Viewer Window which is why I always have a Device connected.



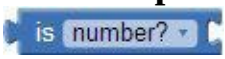
Now we'd better go back to **Blocks** and create the reset Button and perhaps create something to stop being able to get an error if there is no number in the input box.
Drag the **Output_Answer** block out of the **Calculate_But**.Click block. Then go to **Control** and

find the 

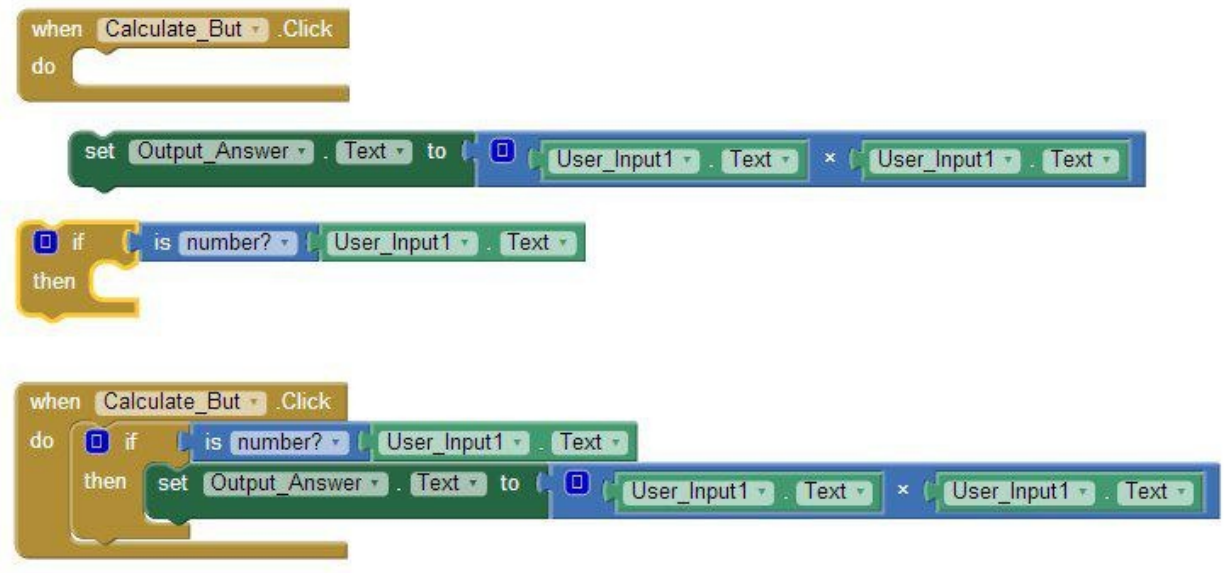
(if then) block and place it in the **Calculate_But** opening. Now go to the **Math** and find the 

(is number?) block

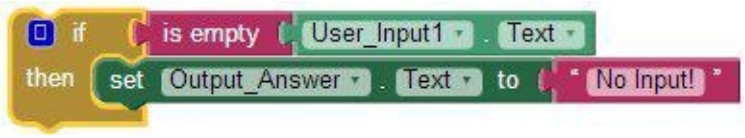
Drag the **Output_Answer** into the "then" opening on the 

block. Now put your cursor on one of the **User_Input1**.text blocks and right click and **Duplicate** it. Take the Duplicate **User_Input1**.text block and put it in the opening on the 

block.



It should look as Above.
What have we done? We have created a process that when you touch Calculate it looks at the User Input(**User_Input1**) and checks, is it a number? If yes it will continue and work out the Output by Multiplying the User Input by itself. If it fails the "**If**" question then it will not continue.
If you touch Calculate now it will only do something if a number is input in the box. And if nothing is there then it will do nothing.
Is that good enough? Well it would be for this because it would be obvious we haven't put an Input in for it to use. Sometimes you have more than one screen of inputs so you'd want to know if all the input boxes have a number so we add another piece of the Jigsaw.

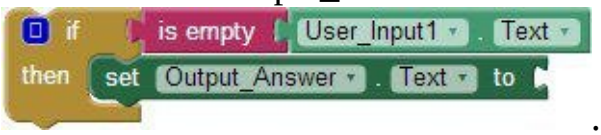


Right we need another 

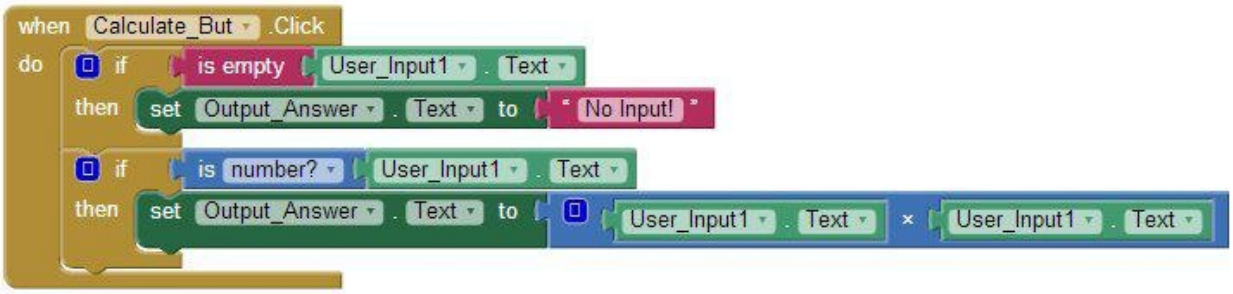
block from **Control**, then in **Text** get 

(is Empty)

block and place it in the top "if" opening of the "if then" block. 

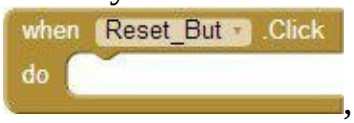
.
Go to a "User_Input1.text" block and Duplicate it and add that to the "is Empty" block.
Go to a "Set Output_Answer.Text" block and Duplicate it and add that to the "then" opening 

. One more piece and were done for this part. Go to "Text" and drag  to the "to" opening from "Set Output_Answer.Text" block.
Now all you have to do is assemble the blocks, grab your new "if Then" block assembly and position it on top of the original one. The dimple should highlight, let it go and it should look like this. If it ends up below the origin it will still work. Now just click in the empty text area and Type No Input!

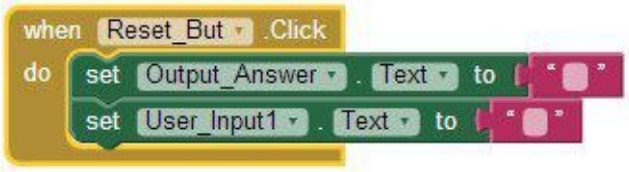


It should now look like this and when you try on you Device, if you just touch Calculate it will show the "No Input!" message in the Output_Answer box. If you input a number it will work it out.
We could leave it there but we might want to reset the App, with only one input and one output you might think it a waste of time. But it's the same no matter how many inputs and outputs there are just more blocks.

Reset Button

We already created **Reset_But** in Designer so it already exists in the Blocks window, so click on **Reset_But** and select "When Reset_But.Click" 

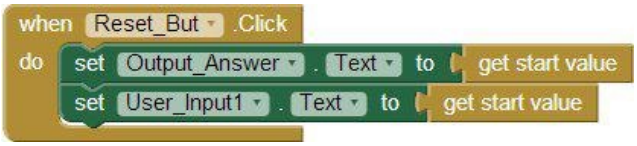
, and now either Duplicate "Set Output_Answer.Text" and "Set User_Input1.Text" or get them from the Blocks window. One more piece and we assemble. Go to "Text" and drag  to the block. Either Duplicate it or get it again and put that one in the "to" opening from "Set User_Input1.Text" . Now put "Set User_Input1.Text" into "When Reset_But.Click" do opening and slide "Set Output_Answer.Text" in below it.



So what does it do? It sets the Input and Output value to nothing or null so you will see a blank space in each.

There is another way to do this, Go to "Control" and drag 

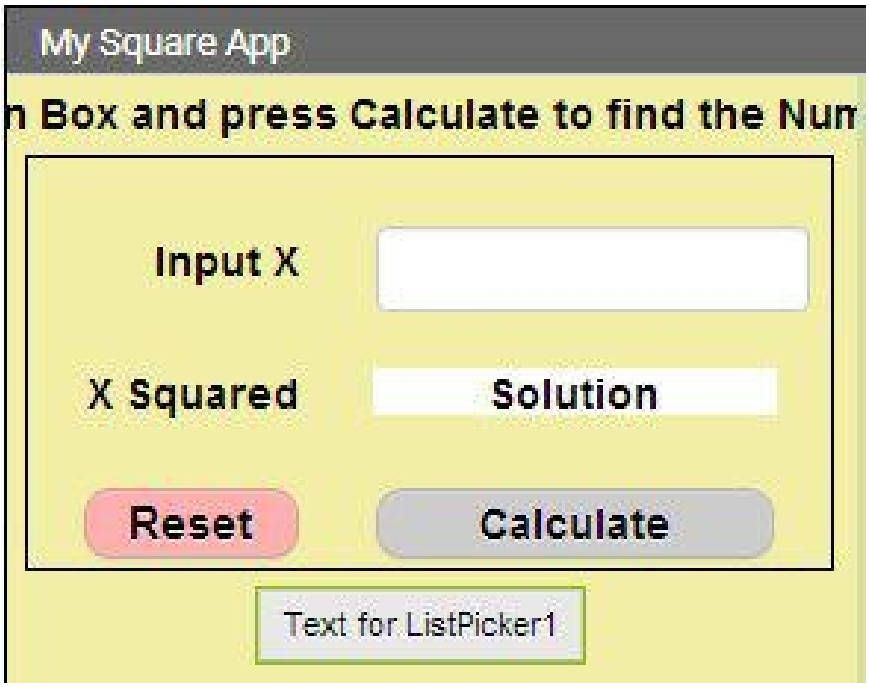
to the block. Again either Duplicate the block or get it again. This time you will get a blank space in output but in our input it will show the **Hint**.



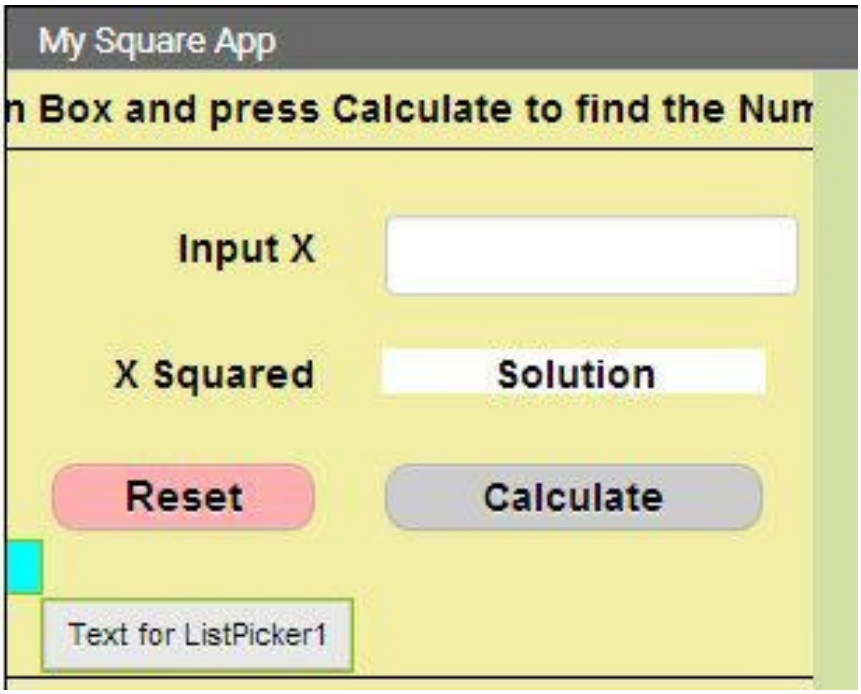
Both will give us similar results but sometimes one will work better and the other won't, so remember there's often or than one way to do most things.

Decimal Places

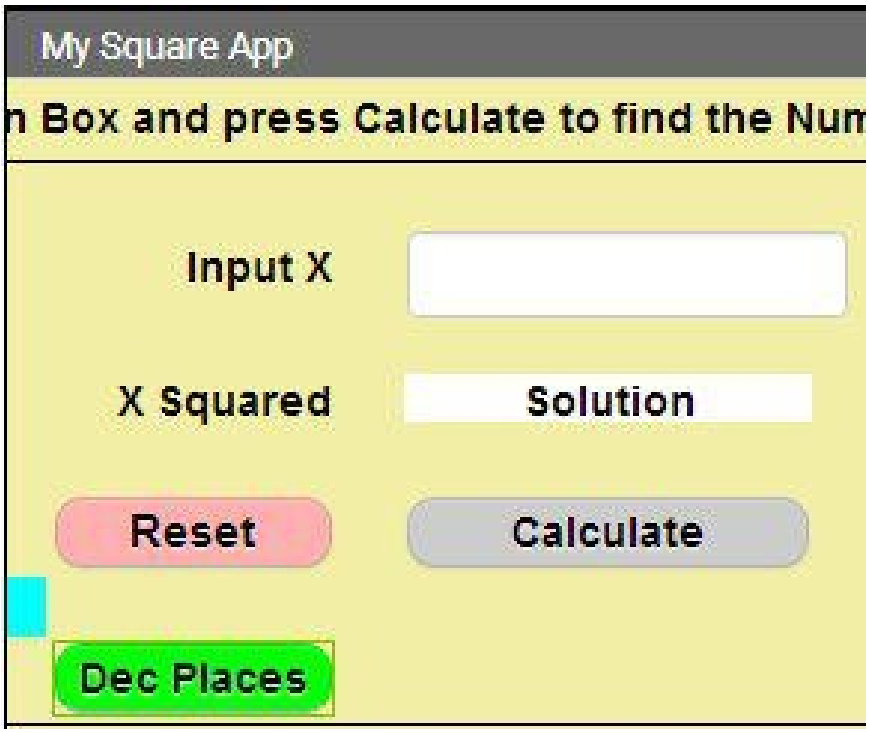
Is there anything else we might want? Yes we might want to change the number of Decimal places in the output. Again it's not so important to this App but on others it could be.
Back to the Designer window then and drag **ListPicker** across to the Viewer window, but where to put it? It will be right against something else. Well for now drop it below the **TableArrangement1** Box.



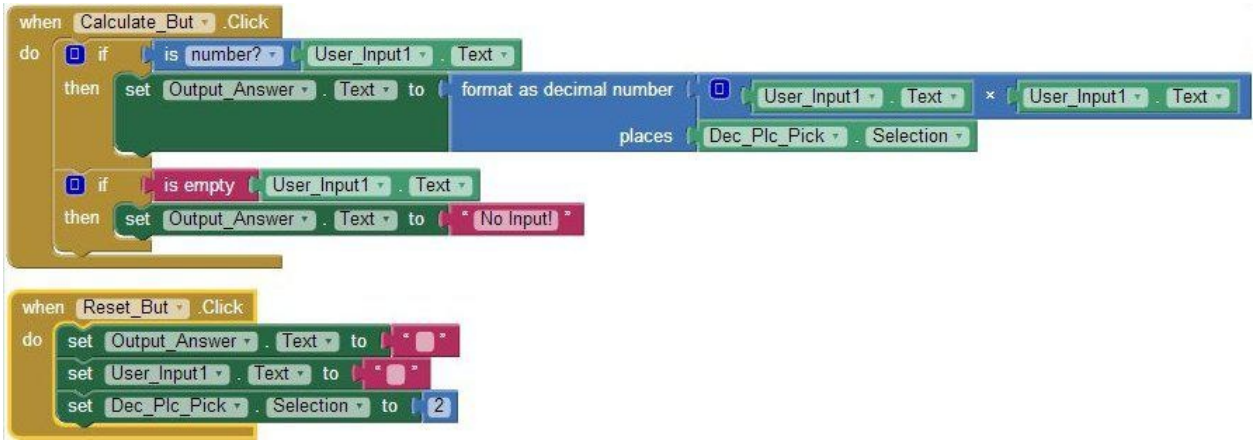
The system is nothing if not flexible, click on either the **TableArrangement1** box or **TableArrangement1** in the Components list, then in **Rows** change 6 to 8. I now put a **Label** in the row below the Reset and Calculate buttons and put a **Height** of 20 pixels and delete the **Text**. I have change the **Backgroundcolor** colour to Cyan so you can see it. Now drag the **ListPicker1** up to the bottom row of the table under the Reset Button.



Again change the **ListPicker1** name using Rename(Dec_Plc_Pick) and make the font bold and 18 and give it a different **Backgroundcolor** and some new **Text**, I use Dec Places, and change shape to Rounded. You'll also need to fill out **ElementsFromString** I used 0,1,2,3,4,5,6 and **Selection** 2, so it's default will be 2.



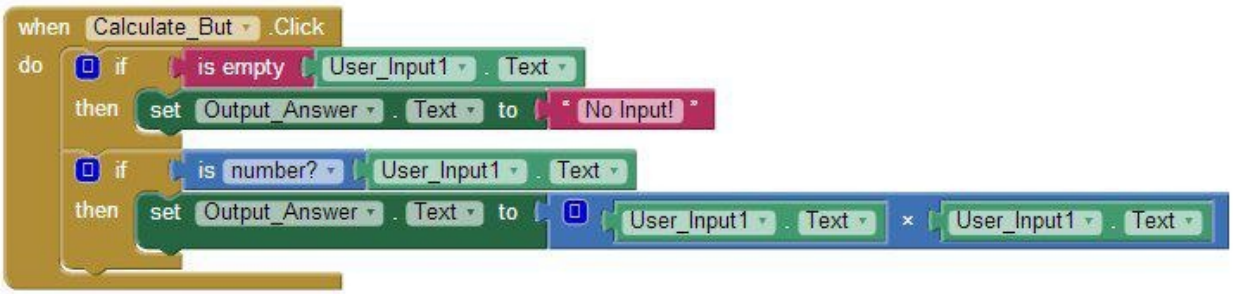
Same old back to **Blocks** screen.
Below is what we want to end up with



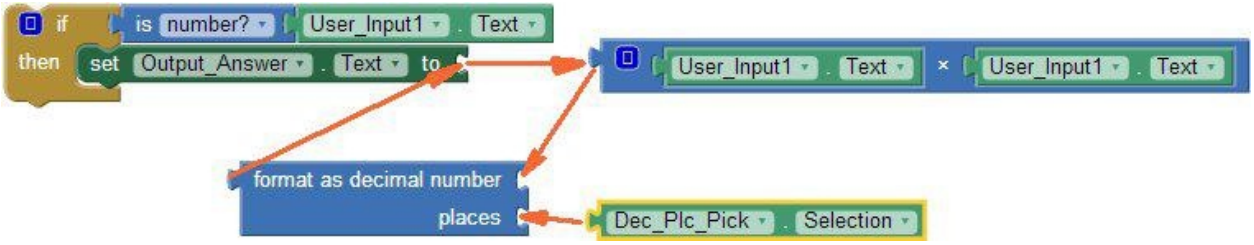
Have a go you'll find the **ListPicker1** blocks in the **Screen1** list and the "format as decimal" and "number" in the **Math** list.

How I did It

Starting the main Calculation as seen below

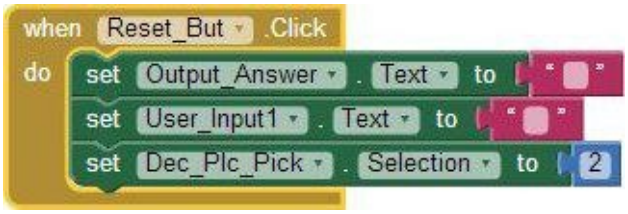


I dragged the **User_Input1** multiplication group of blocks from the **Output_Answer** block.



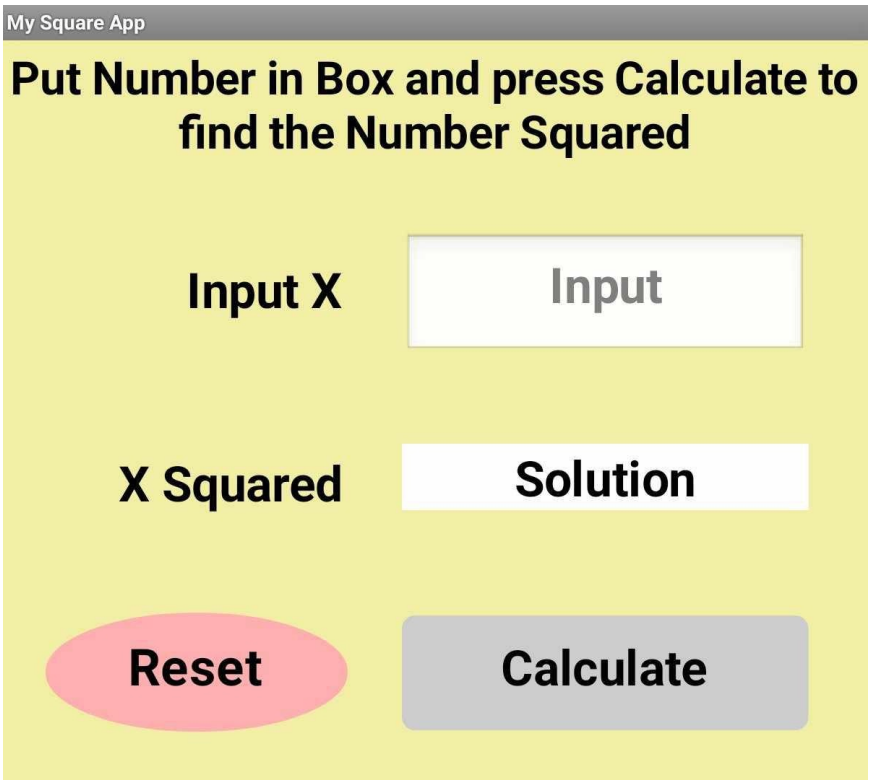
Then went to **Math** in the Blocks window list and scrolled down until I found the "format as decimal" Block and dragged that back to fit in the opening of the **Output_Answer** block, Then I dragged the **User_Input1** multiplication group of blocks and plugged those into the top ("number") opening of the "format as Decimal" block. Then I looked through the Screen1 list in the Block window until I found Dec_Plc_Pick, then clicked and scrolled down until I saw the **Dec_Plc_Pick.Selection** block and dragged that on to the Viewer window and plugged it into the lower ("places") opening of the "format as Decimal" block.
Finally I dragged the If - Then back into the **Calculate_But** where it came from.

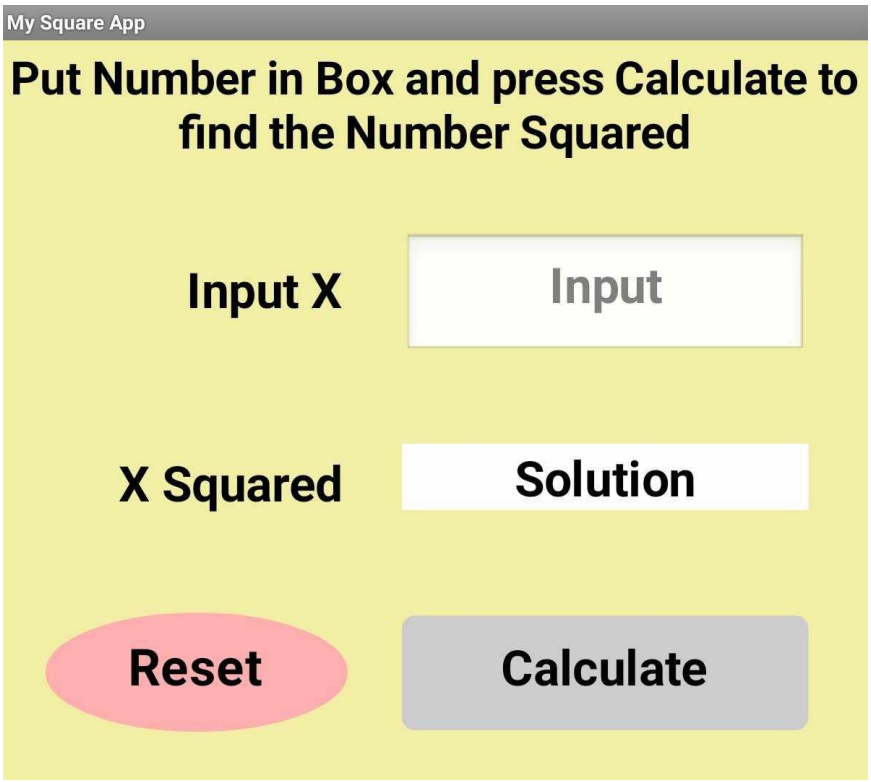
The reset button is even easier.



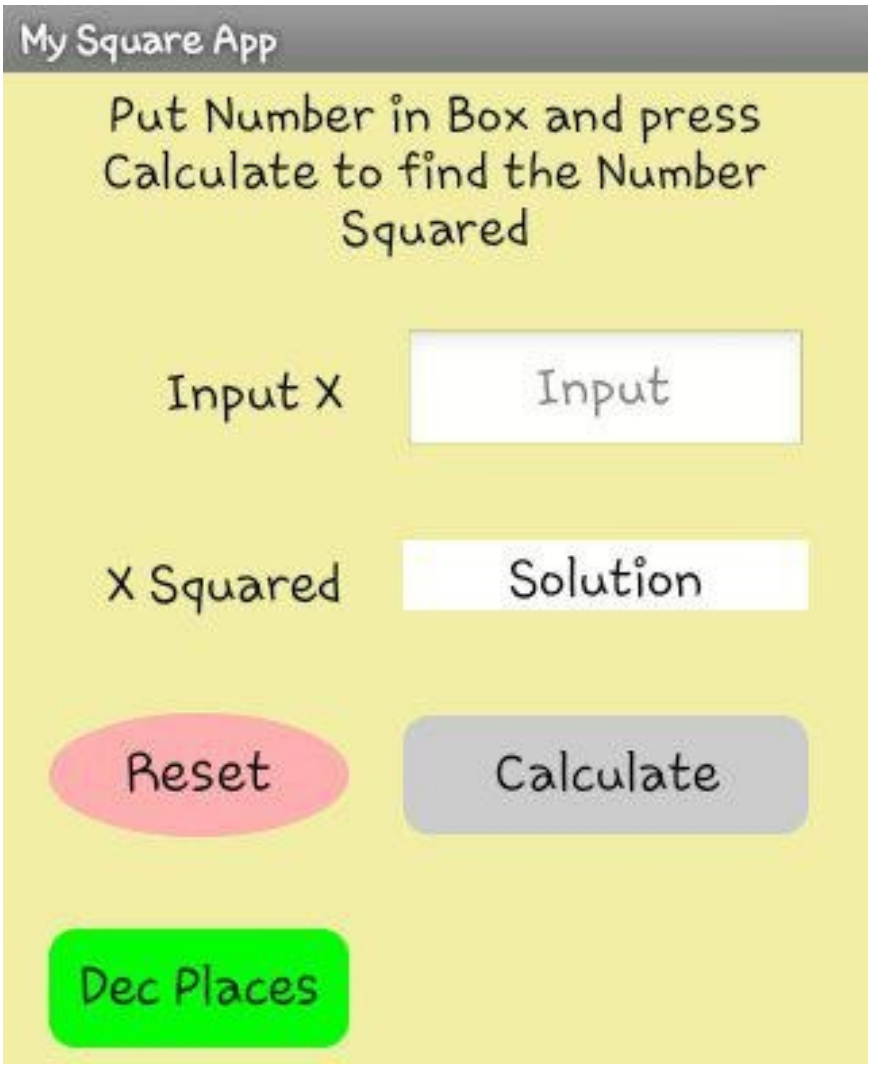
I went to Screen1 list in the Block window until I found Dec_Plc_Pick, then clicked and scrolled down until I saw the "**set Dec_Plc_Pick.Selection**" block and dragged that on to the Viewer window and plugged it in The Reset_But block with the others. I then went back to **Math** list in the Block window until I found number Block(top one), and dragged that on to the Viewer window and plugged it into the open "to" slot.

So we now know what it looks like and were happy

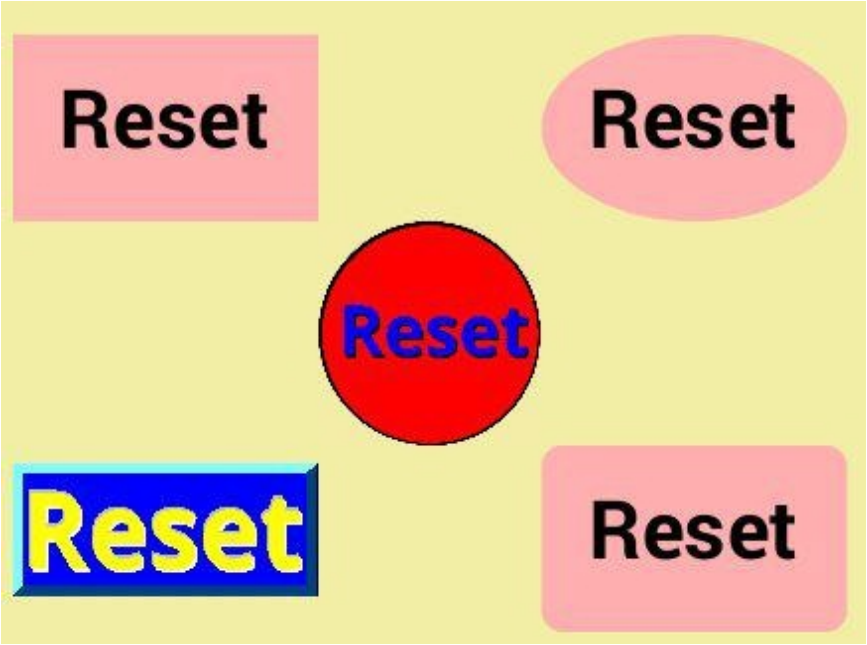




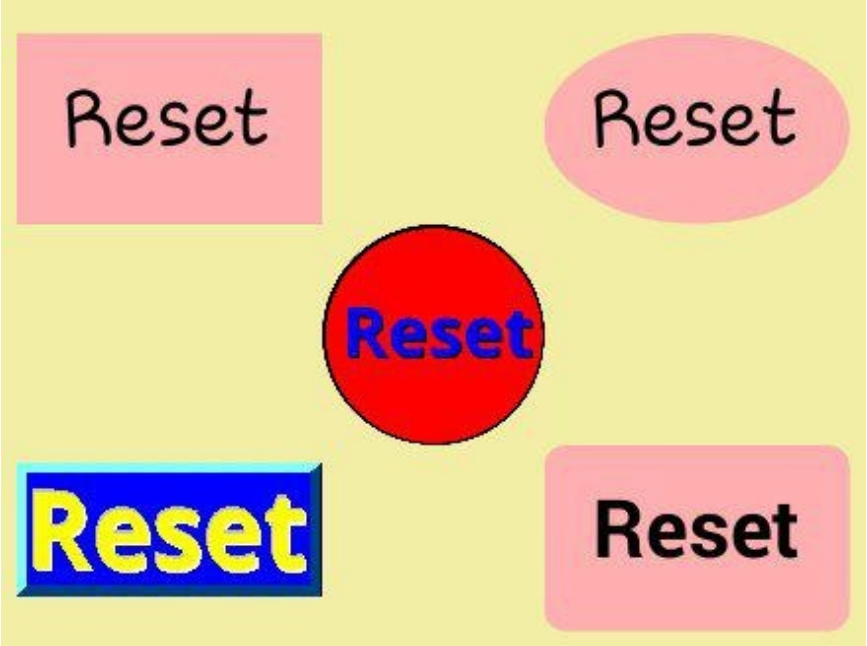
The first image is on a Hudl2 and the second a Samsung Galaxy Phone, they look the same. But this(below) is another Samsung Phone



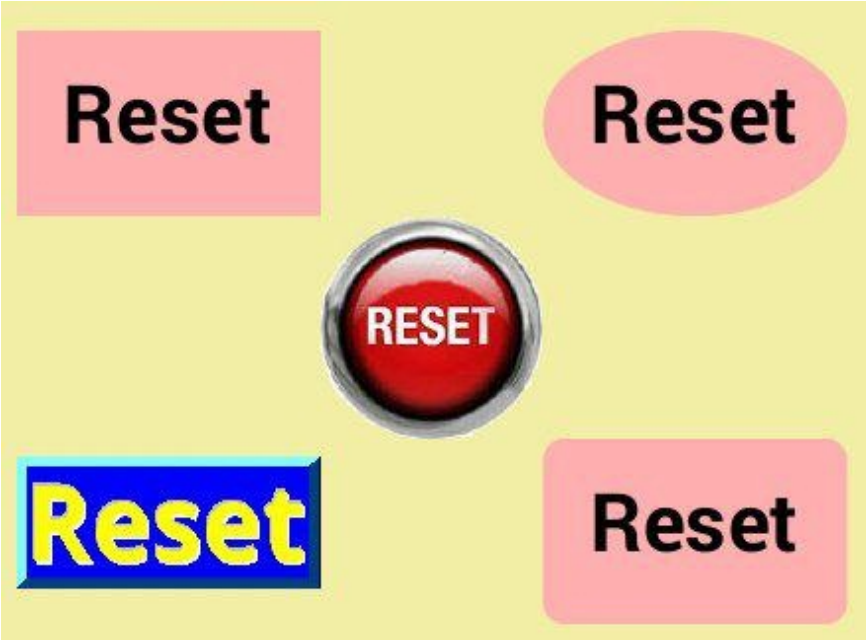
The owner has set their Phone's font style in display to something that might not work with our App or at least change the look and not in a good way. One way around this is in the **Properties** of each component with text change the **FontTypeface** to "san serif", "serif" or "monospace". Another way is to replace the text and use an image.



The top left is a Button with the Text Reset is bold 18 and shape default or Rectangular
The top right is a Button with the Text Reset is bold 18 and shape Oval
The top right is a Button with an Image
The Lower left is a Button with an image
The lower right is a Button with the Text Reset is bold 18 and shape rounded and with **FontTypeface** set to "san serif"

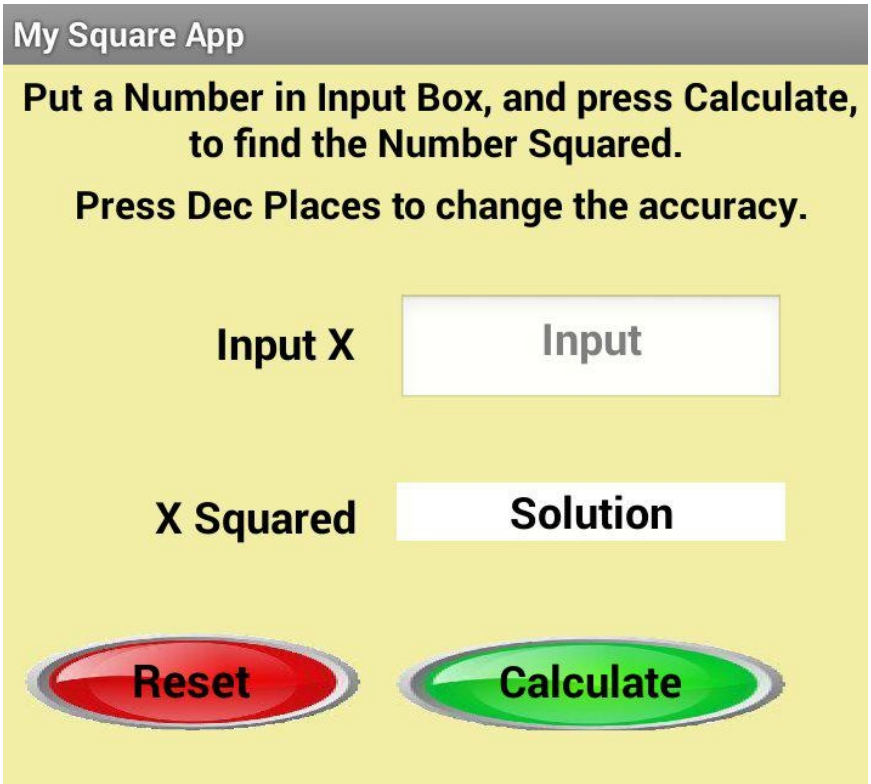


Here they are again shown on the Samsung. As you can see the default text ones change but the Images and defined **FontTypeface** one stays the same.
The Button images I just did quickly in an image editor but you can go on google and type "reset button" and it will come back pages of them.



In this one the centre button is a cropped photo of a reset button, it's totally in your hands how you achieve the look you are going for. It just depends if you have the time and think it's worth the effort of a slick look, the way it works will not change, but if you're going to sell it looks are going to be important.

One final way which I like is to take an image, say a blank circular button then in properties reduce the height to say 40 pixels and then with an 18, san serif font add Text. Have a different colour for each button.



Right, back to the job in hand, are we happy with what we have?

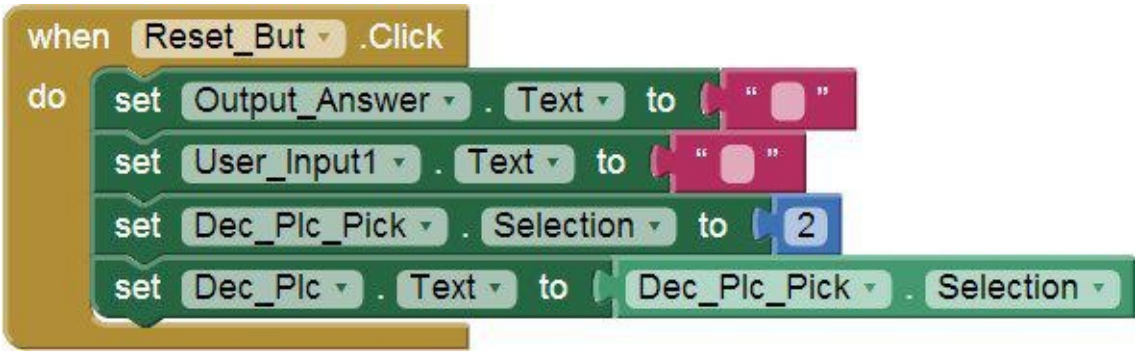
well we could do with an explanation of the "Dec Places" button and maybe an actual figure by it to tell users what it's set to.

So we add a **Label** below the **Txt_Explanation** but above the **TableArrangement1** the **Designer** to give an Explanation of the "Dec Places" button and Renamed it.

Then we add another **Label** in line with "Dec Places" and below "Calculate" and rename it to Dec_Plc. I will put 2 in as text as the default value I created was 2. I will be setting **FontBold**, **FontTypeface** to "san serif" and **FontSize** to 19.0

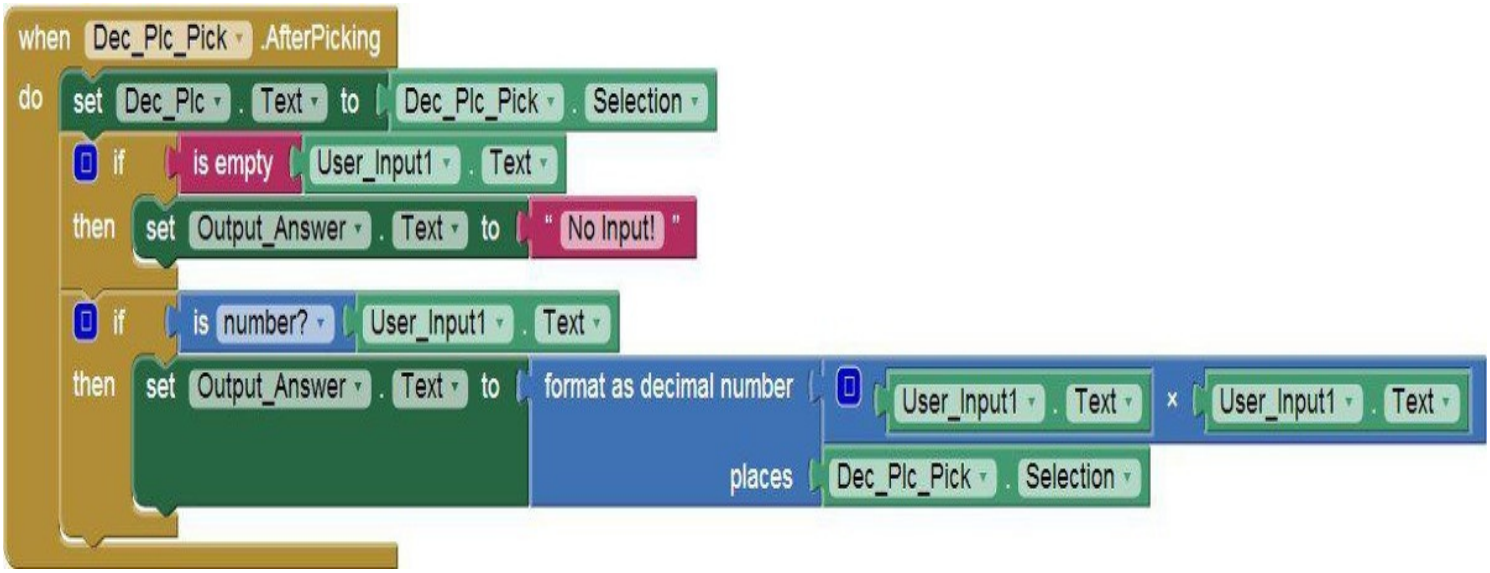
We then go to the **Block** Screen to set up the "Dec_Plc" read out. To start with we find the **label** "Dec_Plc" in **Blocks** and select "**set Dec_Plc.Text**" and drag it to the **Viewer** under our "**Reset_But.click**" block.
Back to **Blocks** and find "Dec_Plc_Pick" and drag out "**Dec_Plc_Pick.Selection**" and drag and drop it to fit in the back of "**set Dec_Plc.Text**" block.

Now drag the pair to the "**Reset_But.click**" block and drop it under(after) the "**set Dec_Plc_Pick.Selection**" block,

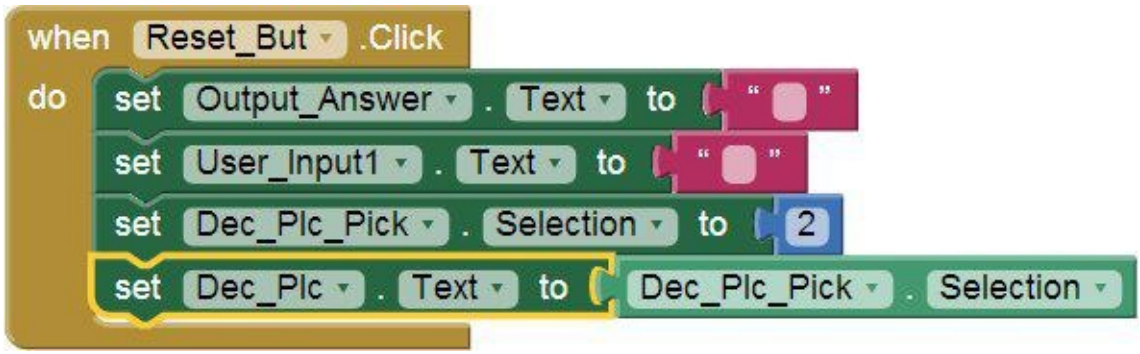


So what does this do, if the user touches the "Reset" button it set the output and input to nothing and "**Dec_Plc_Pick.Selection**" to 2 and now "Dec_Plc.Text" to "**Dec_Plc_Pick.Selection**" value of 2.

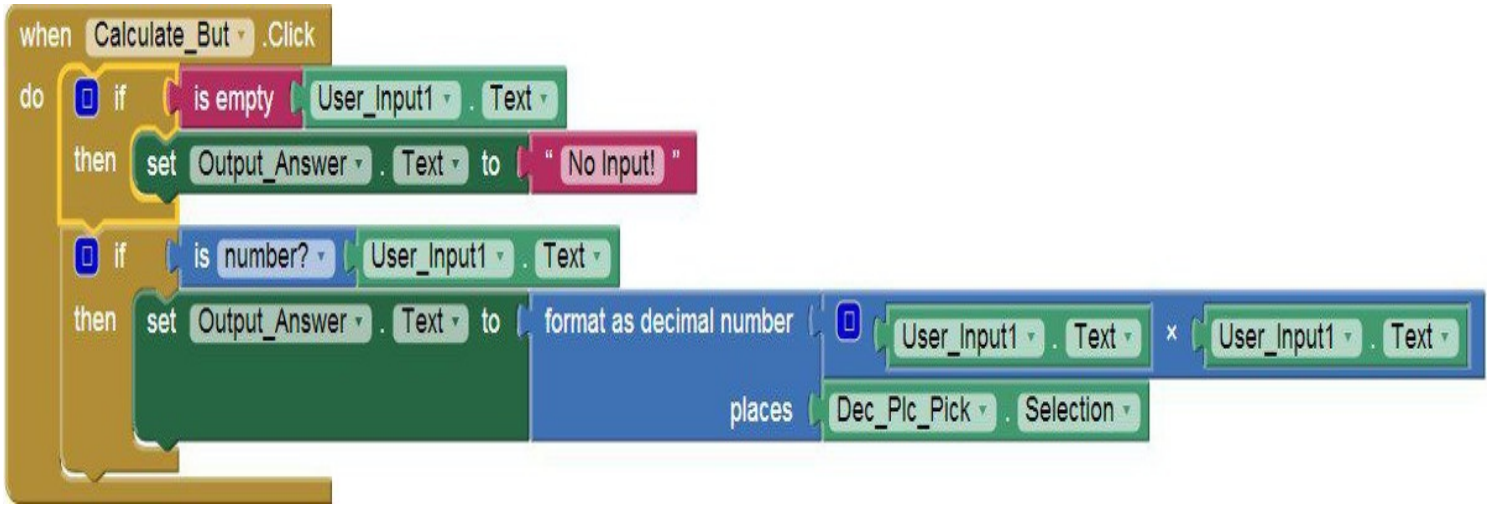
But we still need to tell it to change the output when we change the decimal places, at present to see a change we have to touch "Calculate" after changing the decimal places value.



This looks like a lot of work but all you need to do is get the "**when Dec_Plc_Pick.AfterPicking**" from **Blocks** and find "Dec_Plc_Pick" and drag out "**when Dec_Plc_Pick.AfterPicking**". The rest of it already exist so just find it and duplicate them and put them in the "do" slot.

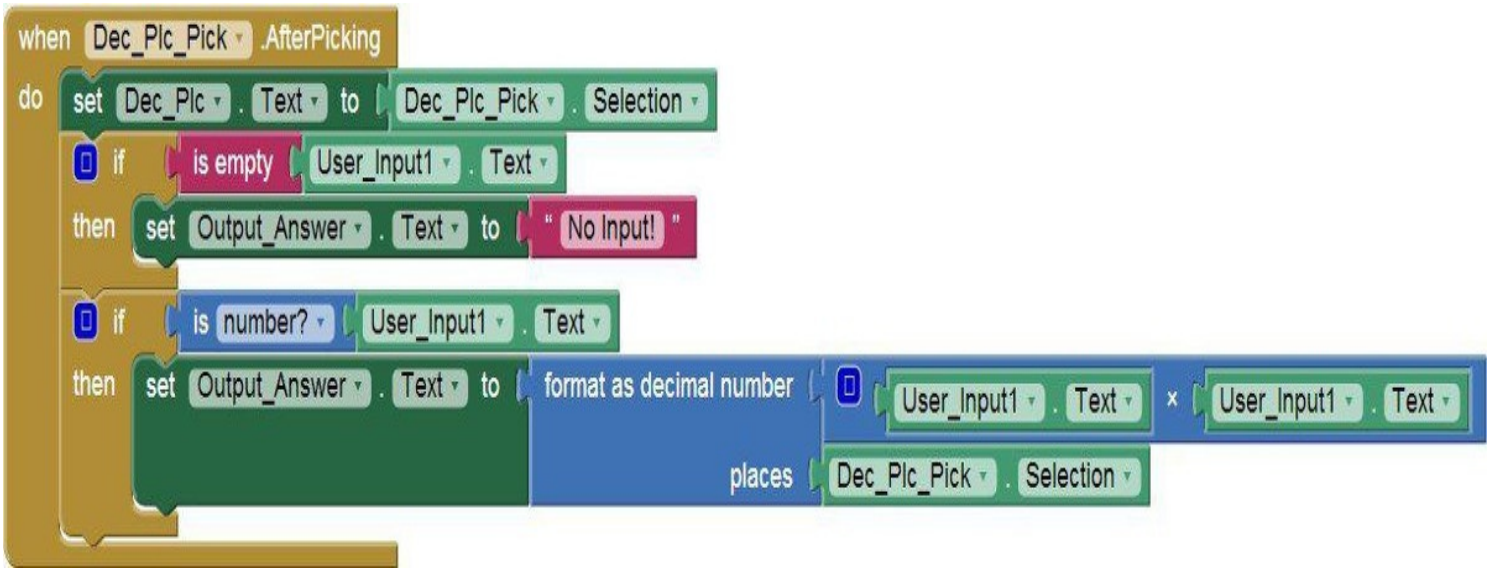


Select the "**set Dec_Plc.Text**" block and duplicate it and put it in the "**when Dec_Plc_Pick.AfterPicking do**" slot.

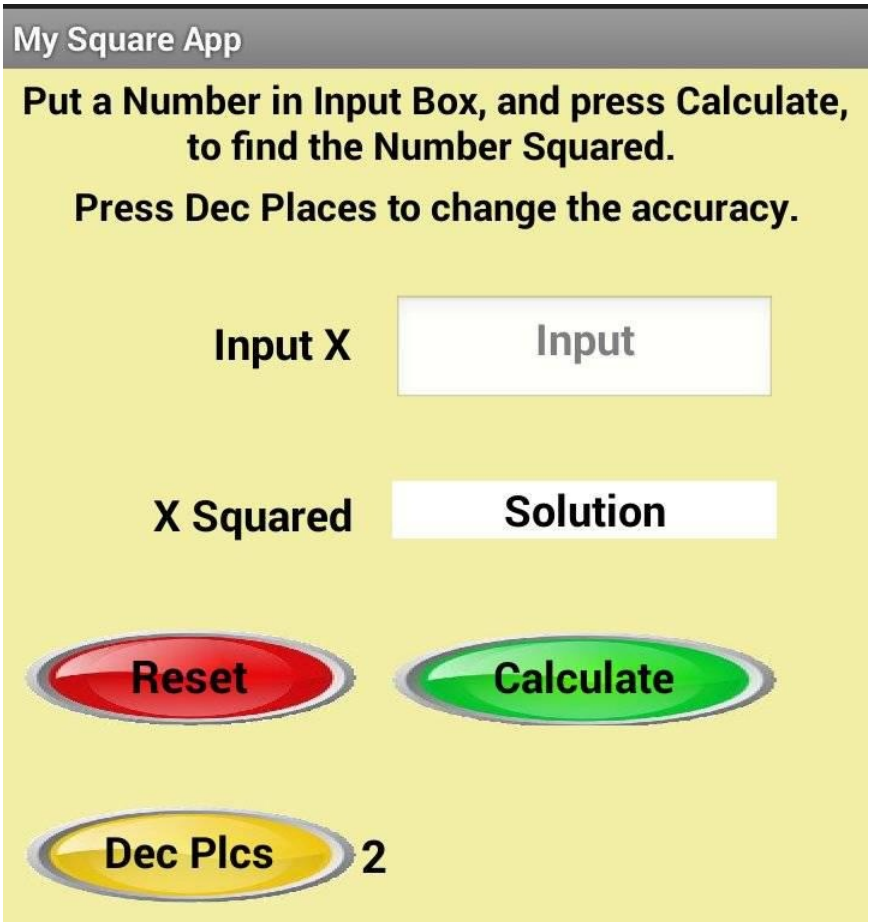


Then select the "if then" block from the "**when Calculate_But.Click**" and duplicate it and put it in

the "**when Dec_Plc_Pick.AfterPicking do**" slot under the "**set Dec_Plc.Text**" block .



And this is what it will look like.



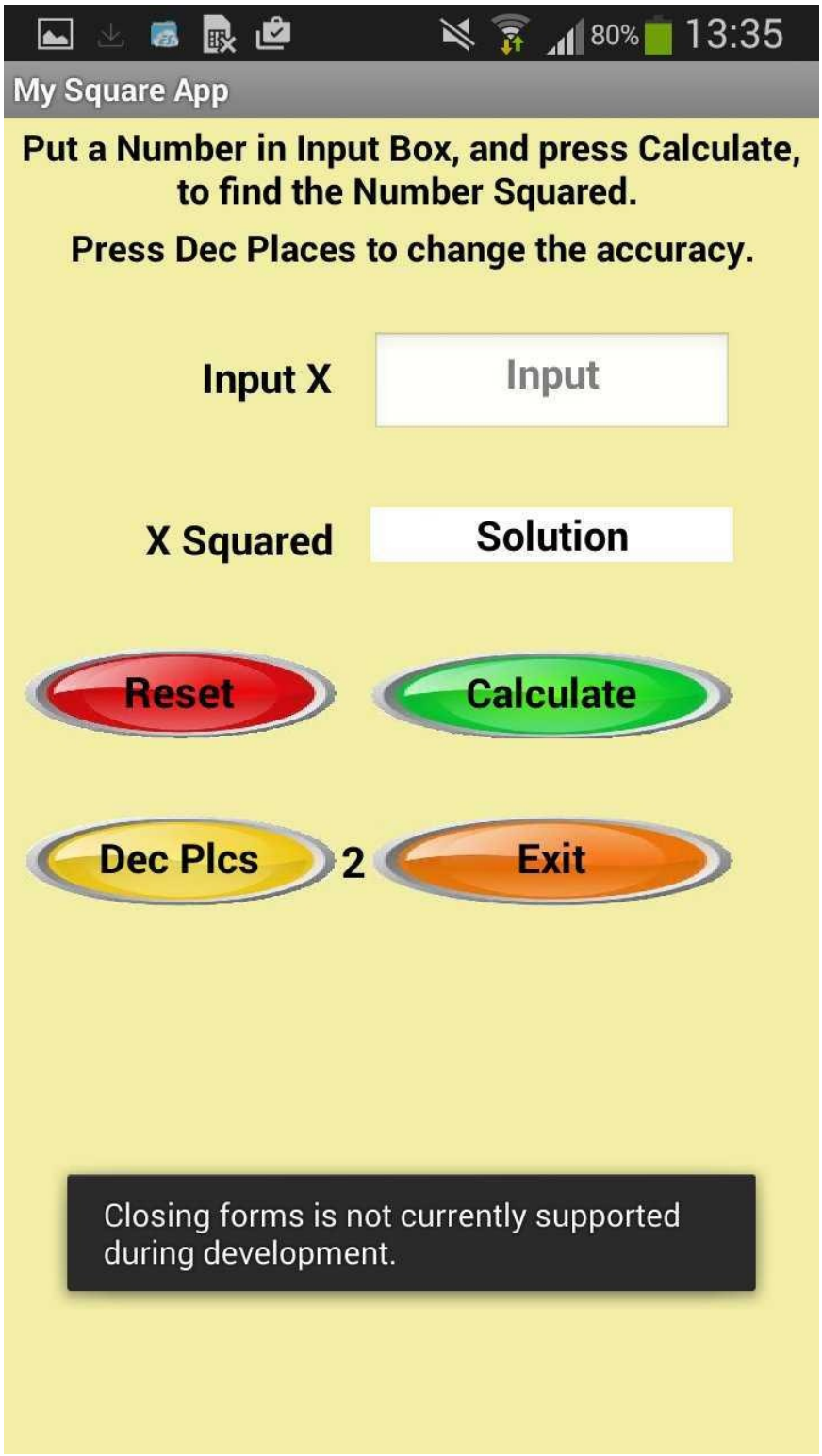
Right do you think that's it for this one? Well no if you look we still have one thing missing to finish this off. A Kill switch, Close, End, Exit, Quit, Shutdown, call it what you want it's much nicer to have a button to leave with. And as with most Apps if you check in Recent or Open Apps it will still appear there until you drag it it across and fully dump it.

This is one of the easier ones, create a button (drag Button from Palette to Viewer in Designer Window) rename it to Exit or whatever 

. Then got to the Blocks Screen and find your Exit Button in Blocks and drag out the "**When Exit_But.Click**" to the Viewer Window. Then again in the blocks window under Built-in look for "**close application**" 

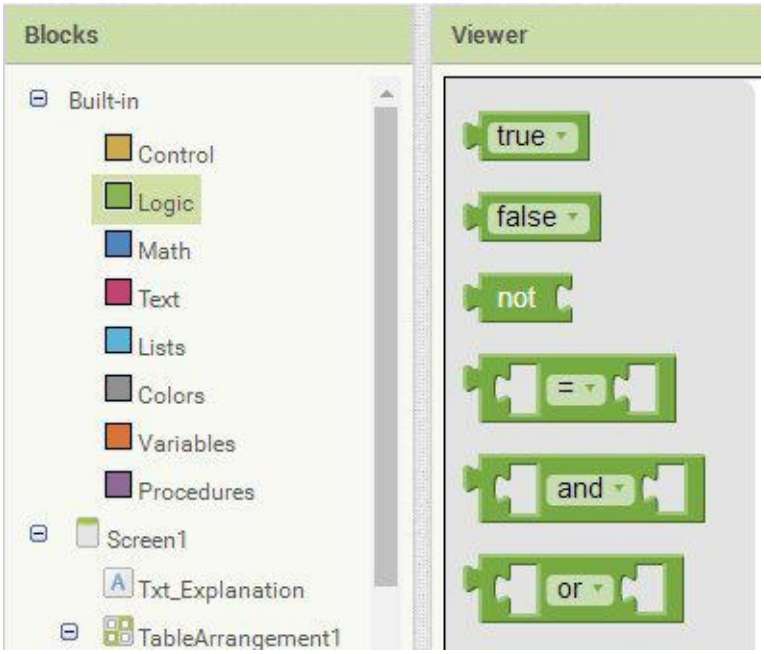
in "Control" and drag that into the viewer window into the "do" slot in the "**When Exit_But.Click**" block.

This what all that looks like on a phone, the text will not vary as we have defined the text used in each case. The message at the bottom is to show what you get if you press Exit during a **AI Companion** session, everything else will work but obviously it doesn't want you to exit the session whilst testing. So the only way to test that is to download it and install it on your device.

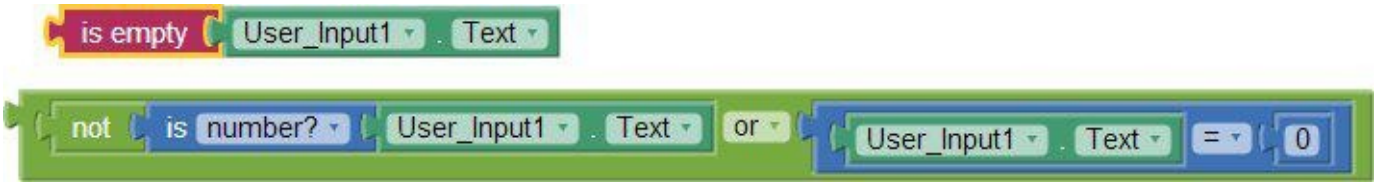


And Or Not

One thing you might find is that in the case of this App the user could put in a Zero so when they Calculate it will find it is a number but the answer is Zero. In this case it's not a problem but in other Calculations a Zero could crash the App so what can we do? we use logic

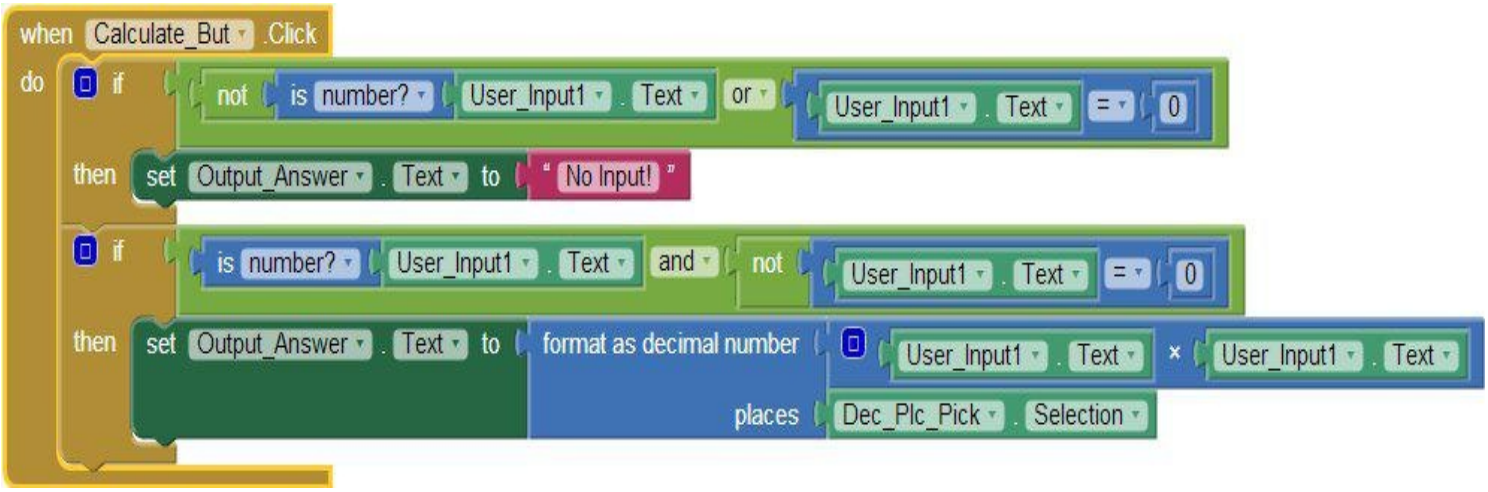


Instead of saying if "**User_Input1 . Text**" "**is empty**" we could say if "**User_Input1 . Text**" "**not**" "**is a number**" "**or**" "**User_Input1 . Text**" "=" "**0**". So if you put that in and then Input Zero when Calculate is touched you'll get the "**No Input**" Text come up.

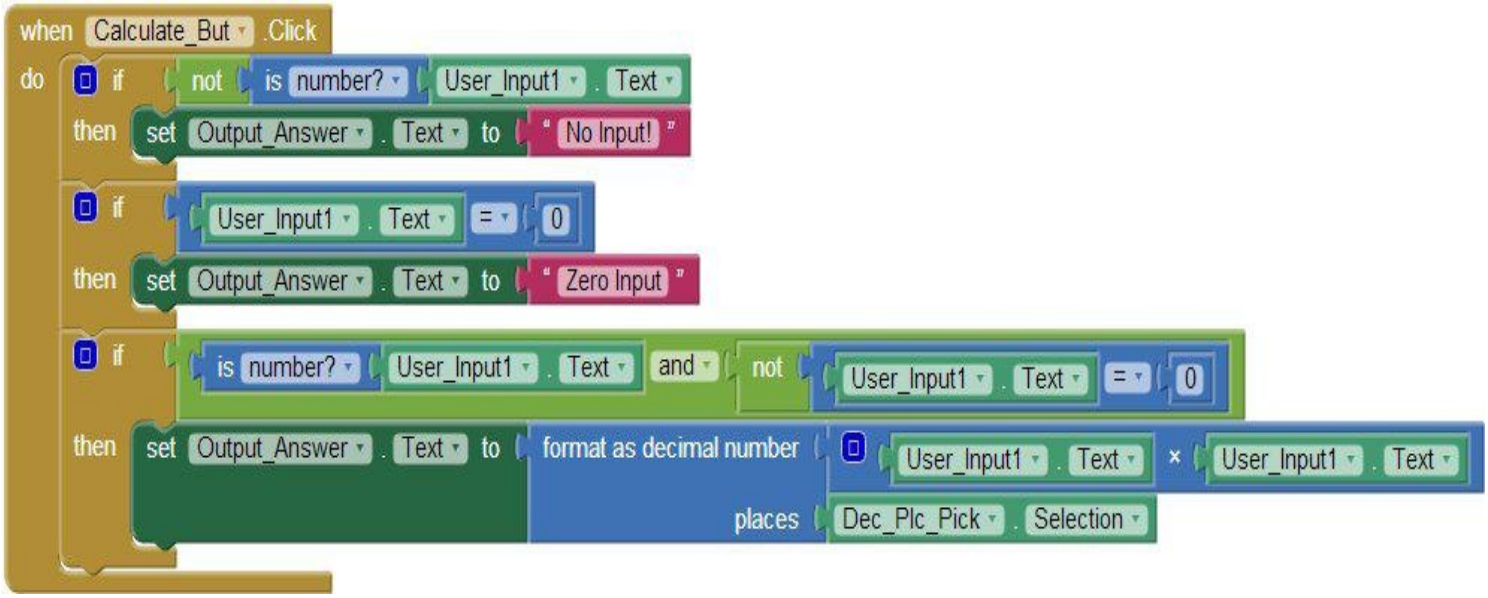


Another way would be to instead of saying if "**User_Input1 . Text**" "**a number**" then calculate. We could say if "**User_Input1 . Text**" "**is a number**" "**and**" "**User_Input1 . Text**" "**not**" "=" "**0**" then Calculate. If we use this and then the user inputs Zero they would just get no solution, but a zero wouldn't cause a crash.

So it's your choice if you want the user to get feed back, put in a check that will tell the user if no number or zero has been entered. Or if you think your user will have more sense, just stop it calculating if a zero has been input. As I said before this is a simple one input Calculation when you have a page or more of inputs it would be advisable to tell the user if something is missing or a number is outside the range it can calculate and why they are not seeing a solution.



You could do both but in this case the "No Input" isn't really true so maybe you'd we to expand the checks to be 2 separate ones with a different message for nothing entered to the one you get for entering a zero?



Even if you are only writing the App for yourself you will probably find error check statements quite useful. In this case it was over the top but if you use it in simple Apps it will be second nature when you come to Apps with 4, 8 or however many inputs and that some must be within a certain range or a positive number or an integer.

You may think why bother with a lot of this, but now you can do this, you can add to it and make it calculate lots of other things. And most importantly you have the basics, there maybe more inputs, buttons, calculations and or outputs but you can do it, most of it you already know.

Example 2

Same basic questions.
How many Inputs will it need?
How many Outputs will it need?
What calculations do we need to do this?

For this example we'll create an App that works out MPG, fuel consumption, that takes 2 inputs Fuel Used (in litres) and distance travelled (in miles) and works out the MPG, Miles per (Imperial)Gallon. In doing that it will convert the litres to Gallons so we could display that and if we've done that we might as well add the distance in KiloMetres, KM, and metric consumption in litres per 100 KM.
Obviously you don't have to follow the example you can have your inputs in whatever units you want and the same with your outputs. But for this example that's the way it will be done.

So in answer to our questions
How many Inputs will it need? 2, Fuel used in Litres and Distance in Miles.
How many Outputs will it need? 1, we need to output MPG. But as we need to convert litres to Gallons so we can work out MPG we can output the Gallons, and if we do that we may as well convert miles to KM and MPG to litres/100km. So we will have 4 Outputs.
What calculations do we need to do this? We need 2 but will want 4.
1, litres to Imperial Gallons - Litres × 0.219969
2, MPG - Distance Travelled(in Miles) ÷ Fuel Used(in Imp. Gallons)
We don't need the following 2 but we can include them.
3, Miles to Km - Miles × 1.609344
4, MPG to Litres per 100Km - 282.481 ÷ MPG

It's probably worth saying at this point if you're working out something for the first time don't just use the first equation, or numbers you find on the internet. They can be wrong. Use books, or a number of sources and work it out a different way if you can, to double check. I found most internet site use 22/7 (3.14285714) as Pi instead of 3.14159265 not much difference at first glance you might think but when it's squared or cubed errors get big quick. Check and double check!

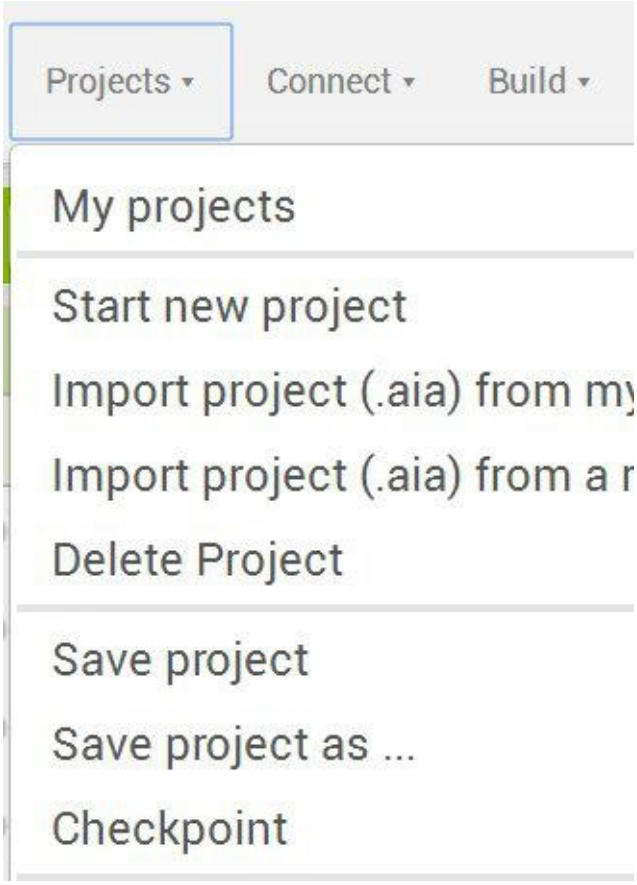
$\pi r^2 = 314.159$ $\pi r^3 = 3141.592$

$22/7\ r^2 = 314.285$ $22/7\ r^3 = 3142.857$

Using 10 as a radius the differences start getting into tenths and then whole numbers and that just with the most basic of calculations.

Initially we'll create Example 2 by doing the minimum to Example 1, to make it work. So open that and go up to Projects and click on **"Save project as..."**, I called mine Example_2_mpg.

Again don't get hung up over names this is the file name, not the final App name.



If we go to the **Designer** screen and in select **TableArrangement1**. Then look at it's **Properties** you'll see we have 8 rows so you can increase that to 10. Or Put in a second **TableArrangement** under the first. You might wonder why but as you will see to add another input in the **Viewer** we will need to move all the Components below that point down. This is a simple App so this is not a big job but once you start on bigger Apps it will pay to have separate **TableArrangement**s for Inputs, Outputs, Buttons and as you'll see later, Variables.

Ok you've added 2 rows (or a new **TableArrangement**) so now you have to select each block in the **viewer** and move them down 2 spaces until you have 3 rows between the Input and Solution. Don't forget the spacers you put in, it's easy to drop one block on top of another without realising it. We want a **Label** in the cell in the middle of the input and output text(**Label**s) and a **Textbox** in the cell in the middle of the input and output **Textbox**es. We also need a spacer between our new row and the Output row.

Next we need to rename and change Text in the display so it makes sense as a MPG Calculator. We'll need
Miles as input text and a Mile_Input as our **Textbox** name in our 2nd row. The Hint can change to Miles or Mile Driven.
Fuel (Litres) as input text and a Fuel_Used_Input as our **Textbox** name in our 4th row. The Hint can be Fuel in Litres.
MPG as output text and a Output_Answer can stay as our **Label** name in our 6th row.

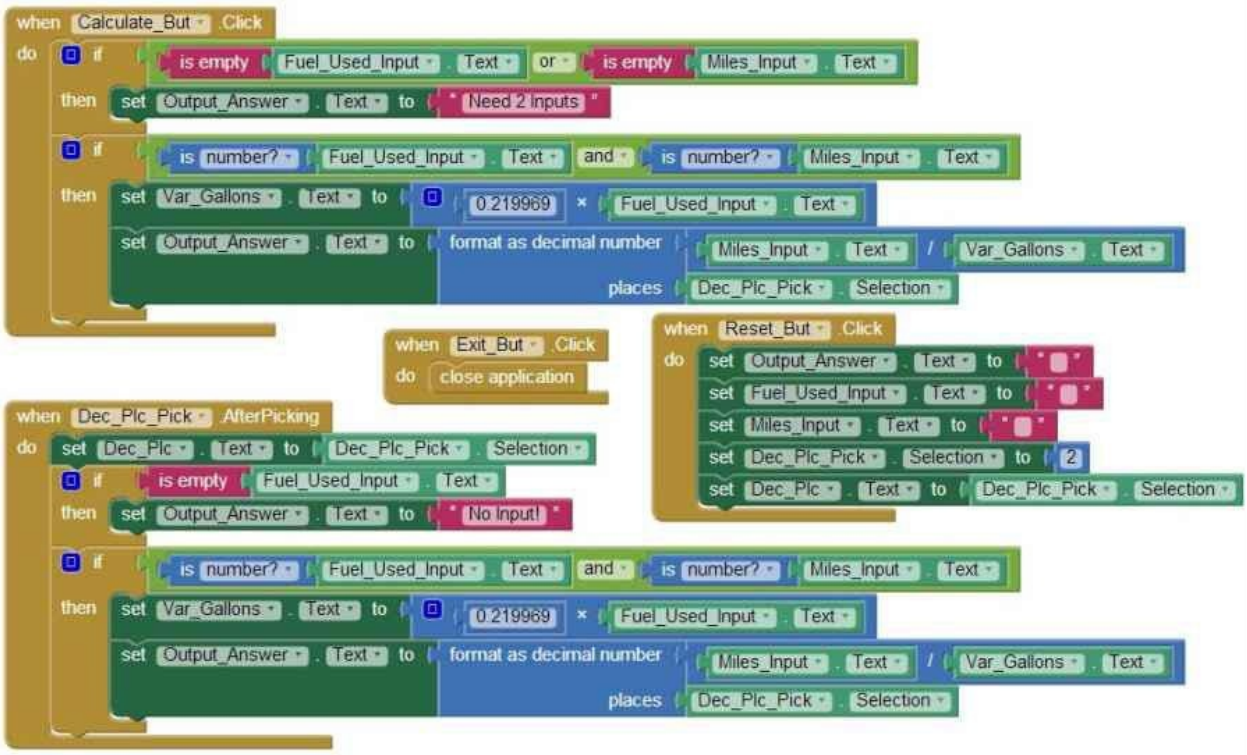
While were doing this we might as well click on **Screen1** in **Components** and change:
AppName to MPG Calculator or something
AboutScreen to This App will work out MPG (Miles Per Gallon). Given Distance in Miles and fuel used in Litres.
Title to MPG Calc App or whatever you like.

Initially we will only work out the MPG, we still have to work out litres to Gallons but won't display it and although that's not really a problem in this example it gives us a chance to see how we can use variables. So for the first part we only need 2 calculations

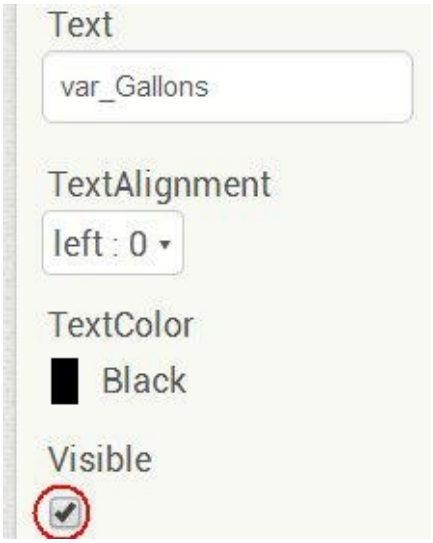
1, litres to Imp. Gallons - Litres × 0.219969
2, MPG - Distance Travelled(in Miles) ÷ Fuel Used(in Imp. Gallons)

As you will see below there more than one way of doing variables.

The obvious one as above is to use **Variables**(orange) in Blocks, on the Blocks Screen. There is nothing wrong with this way of doing it but there is another, which can have advantages.
By now you should have noticed that each type, group or class of block has it's own colour. Inputs (**TextBox**) light green, Outputs(**Label**) dark green and Built-ins, **Math** blue, **Text** red etc. and now **Variables** orange.



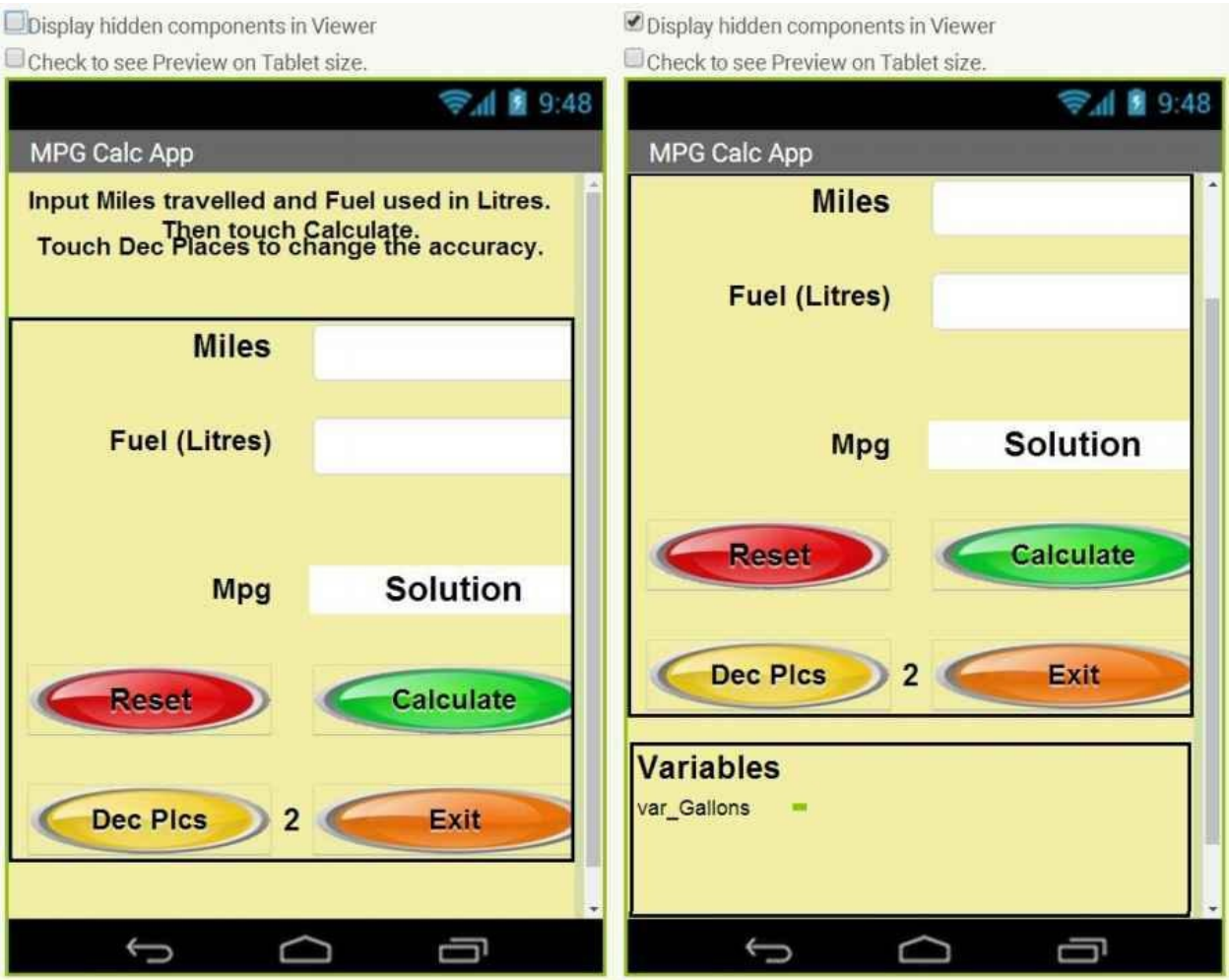
Above is the way I tend to do it, why? In this case it's makes no difference because we intend to use the "Gallons used" as an output later and even if we didn't we could just use "Litres × 0.219969" everywhere we need "Gallons" and as said in this simple example that's fine. But if we had 10+ variables and our final output is wrong how do we track the error back to it's source. Using this second method it's easy because all our variables will be in a table which is invisible. But whilst we are debugging the App we have it visible and can see the values of all the variables as we go. The look of the final App is identical but whenever we are testing it we can make the table visible and see all the variables as we try different inputs. You could create a your variables as labels and then untick the **Visible** box.



This is ok and will work. But you'd need to do this to each variable each time you want to see them. The way I do it, is first to create a new **TableArrangement** and then put all my variables in

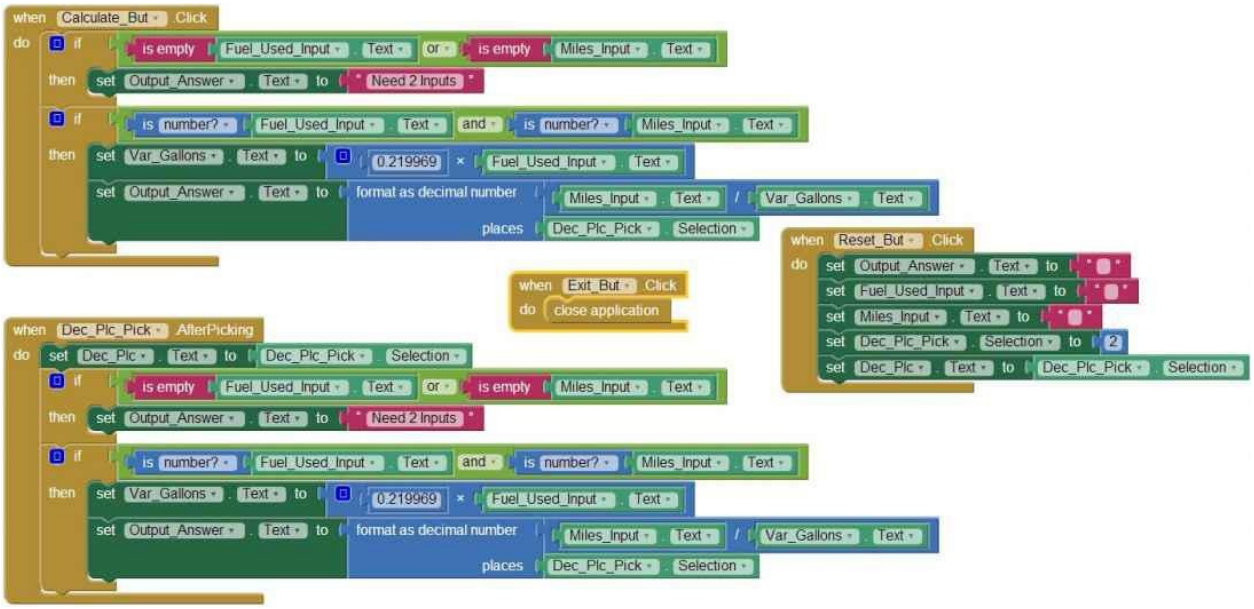
there. This way you untick the **Visible** box and they are all invisible.

Don't be confused in the Viewer window, if you tick "**Display hidden Components in Viewer**".

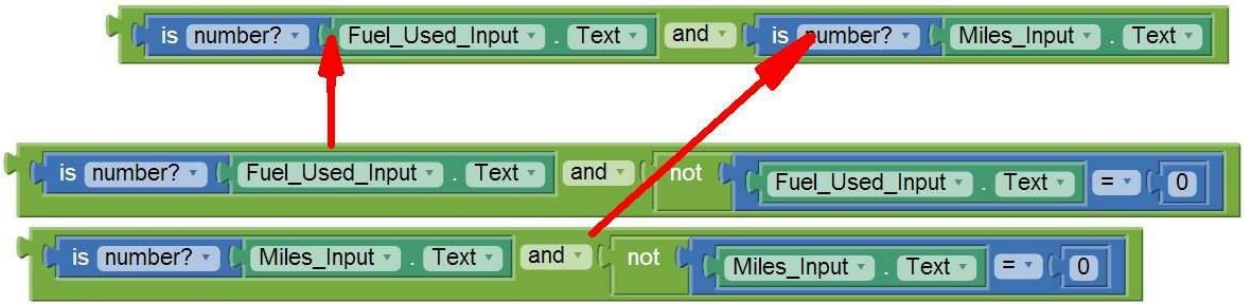


You will see them, in the Viewer window, but not on your Device through Companion or as a downloaded App. To see the hidden items on your Device they must have their properties set to **Visible**.

Back to our example. You may wonder why I'm using "0.219969" and not "0.219969248299" or even "0.22". Well it's up to me or the customer, for this case I'm not going to want it more than 2 decimal places so I could get away with 0.219 but at some point in the future I might want a better accuracy so I've gone with 6 decimal places. Some will say if you know it to 10 places, that what you should use, but I think we are carrying a lot of data and making it work slower. Bottom line is make it more accurate than you think you'll need to and then you're safe.
It's like Pi(π) most calculators have Pi built-in, but if like **MIT App Inventor 2** they don't, what do you use? Some use 22/7 which is ok to 2 dec places also 333/106, which is good to 5 places and 355/113 is good down to 6 dec places and so on 52163/16604...103993/33102 each one getting a bit closer to true Pi. You could use 3.1415926536 again it depends what your expecting out of the App.

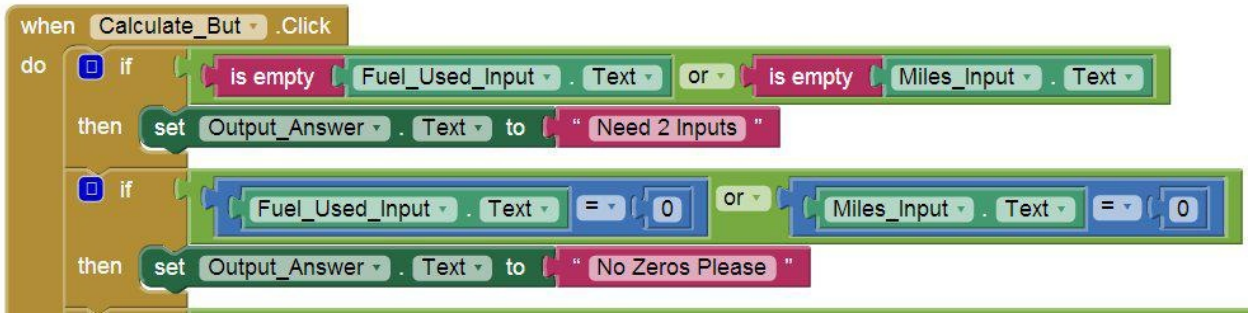


Ok our App works. What happens if suppose we use a value of zero? According to our rules, Zero is a number so it won't cause "Need 2 Inputs" to be displayed. But Zero will crash the App as you can't divide or multiply Zero!



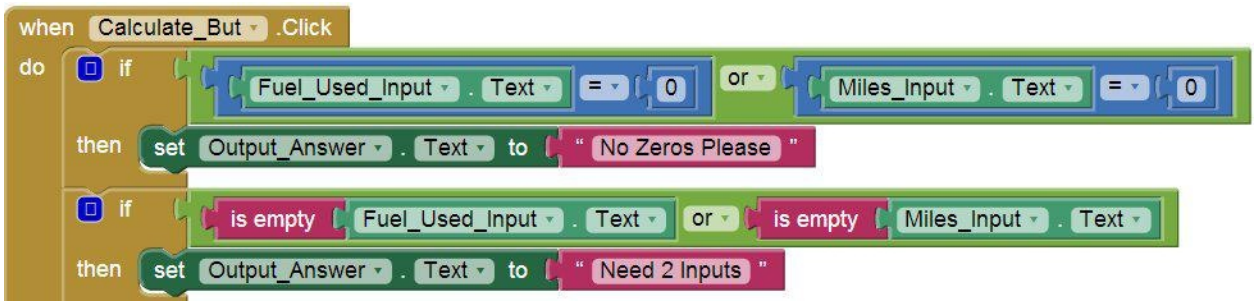
So we need to expand our rules to say if the values are both numbers and not Zero then continue.

We could have change the "if is empty" to "if is empty or equal to 0" and set the Output_Answer to be "need 2 Inputs above 0"



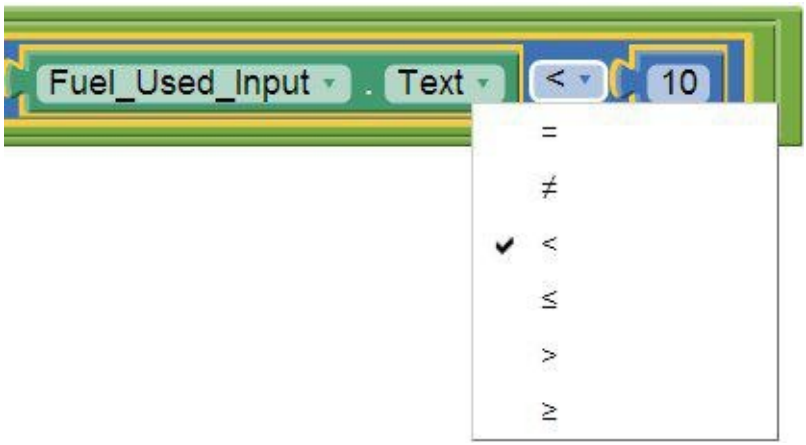
Or put an extra line in, so if either input is a Zero it will say "No Zeros Please". Remember there

is no wrong way unless it doesn't work, but there are loads of different ways that do work. By using the "No Zeros Please" we get a definitive No Zeros if one is input. Now it's in 2 lines or Blocks the order will have some bearing as I've done it, if you put in a Zero and no other input it will come back with "No Zeros Please" had I changed the order of lines/blocks



This will show "Need 2 Inputs". It still fails on the Zeros rule and flashes up "No Zeros Please" but then it fails the "Need 2 Inputs" rule and flashes that up.

For some Apps you will need to have set a range, larger than this and or smaller than that.



You may want or have to limit the Input values to say between 0 and 10. Set your **.text** to equal your value then select the upside down triangle by the equal sign and you are given a selection. Equal, not equal, less than, less than or equal to, more than and more than or equal to.

If you try less than you might then realise you've limited to 0-9.99999999999999 and not 10 so you can use the less than or equal to sign.

Ok back to the job in hand, our App will now;

accept 2 inputs
check they are numbers
check they are not Zero
convert the Litres to Gallons
Calculate the Miles per Gallon

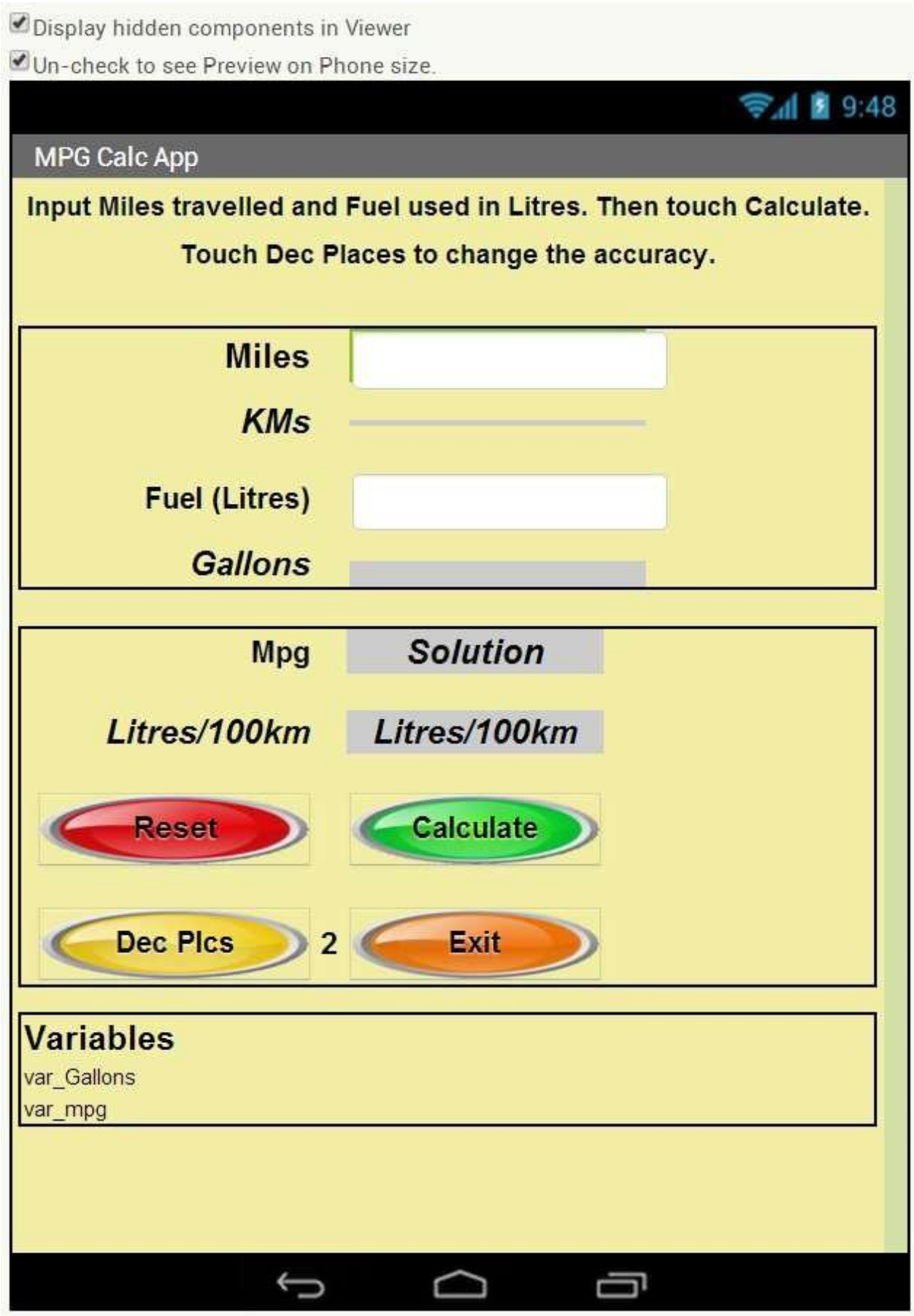
We can;

Reset all the values
Change the number of Decimal Places that are displayed
or Exit the App

We could say that is it and finish here, but we won't, we will also convert the Miles to Kilometres and the MPG to Litres/100km. This is where we will use our Variables again. Why you ask? Well we don't need to in this case but on a larger App you might want to so we will use it here to Practice. The main reason is that when we work out the MPG it will be to X decimal places (set either you or by the user) and if we take that number and then convert it to Litres/100km it will be less accurate than if we used the original numbers. And the same with Gallons if we convert the Litres to Gallons and display that it will be constrained by the user Decimal Places and affect the final MPG Value. We could just use all the Calculations again to work out each value from inputs but that means we are duplicating parts or all of the equations and we could make mistakes. So we will work out a Variable, litres to Gallons and MPG and then use those in our later equations and to create the separate Output Value that has it's Decimal Places set.

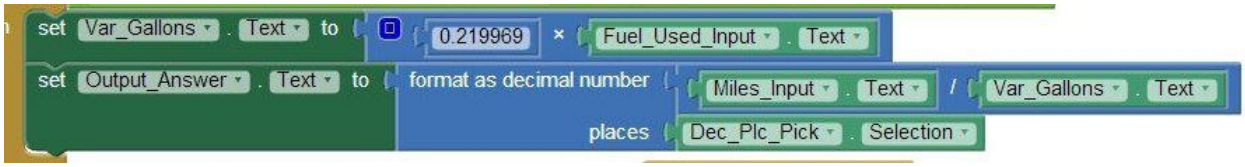
How did we do it?

Back to the Designer Window and add in some Labels for Kilometres and it value, Litres and it's value and Litres/100km and it's Value. You could just put them between the existing Labels or put more rows in and space it all out again.

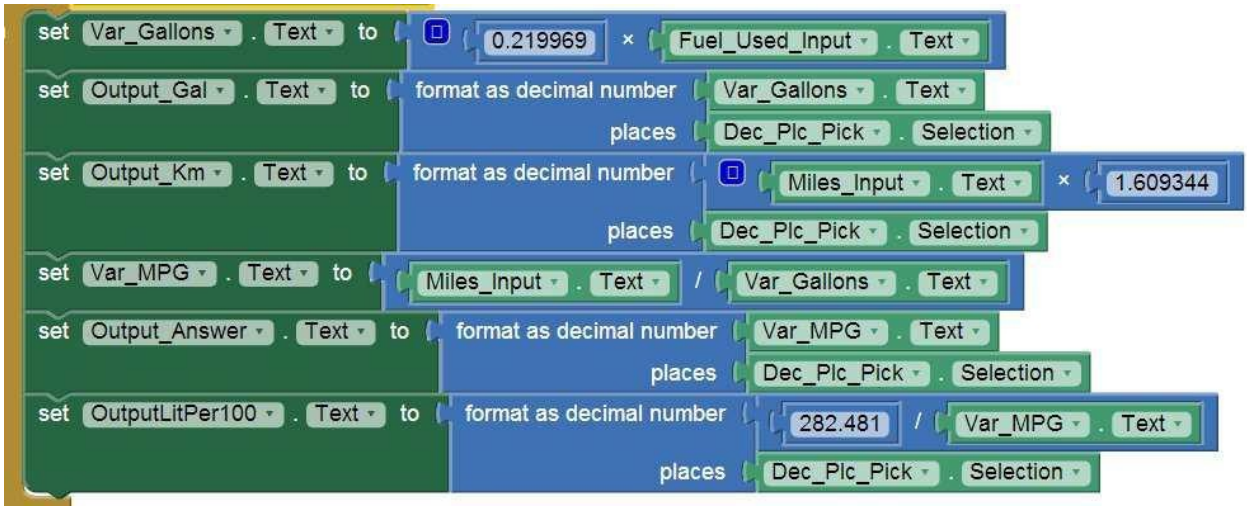


You may notice that I Created a new Table above what was the 1st one, and moved Miles and Fuel (Litres) and their Inputs(Text Boxes) up there and added KMs and Gallons together with two

new Outputs (Labels). I also moved Mpg up and added Litres/100km in the Middle(original) Table. Finally I added Var_mpg (label) to the Variables table. Don't forget to add any spacers you might want to create the right spacing.
Now on to the Blocks Window, this is our original 2 lines of Block Calculations.

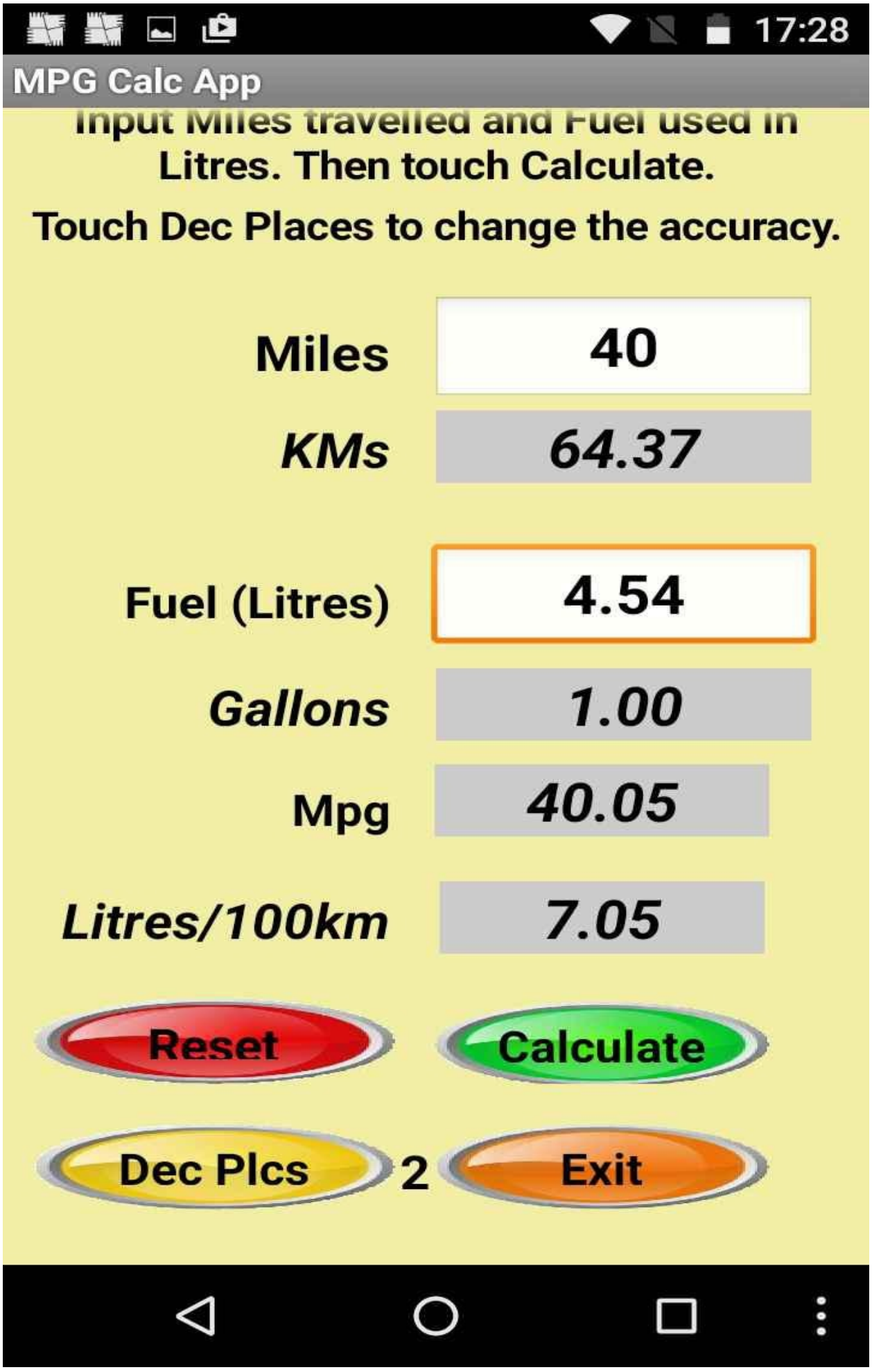


Here's our new workings, it's almost as it was except we've added a few lines.



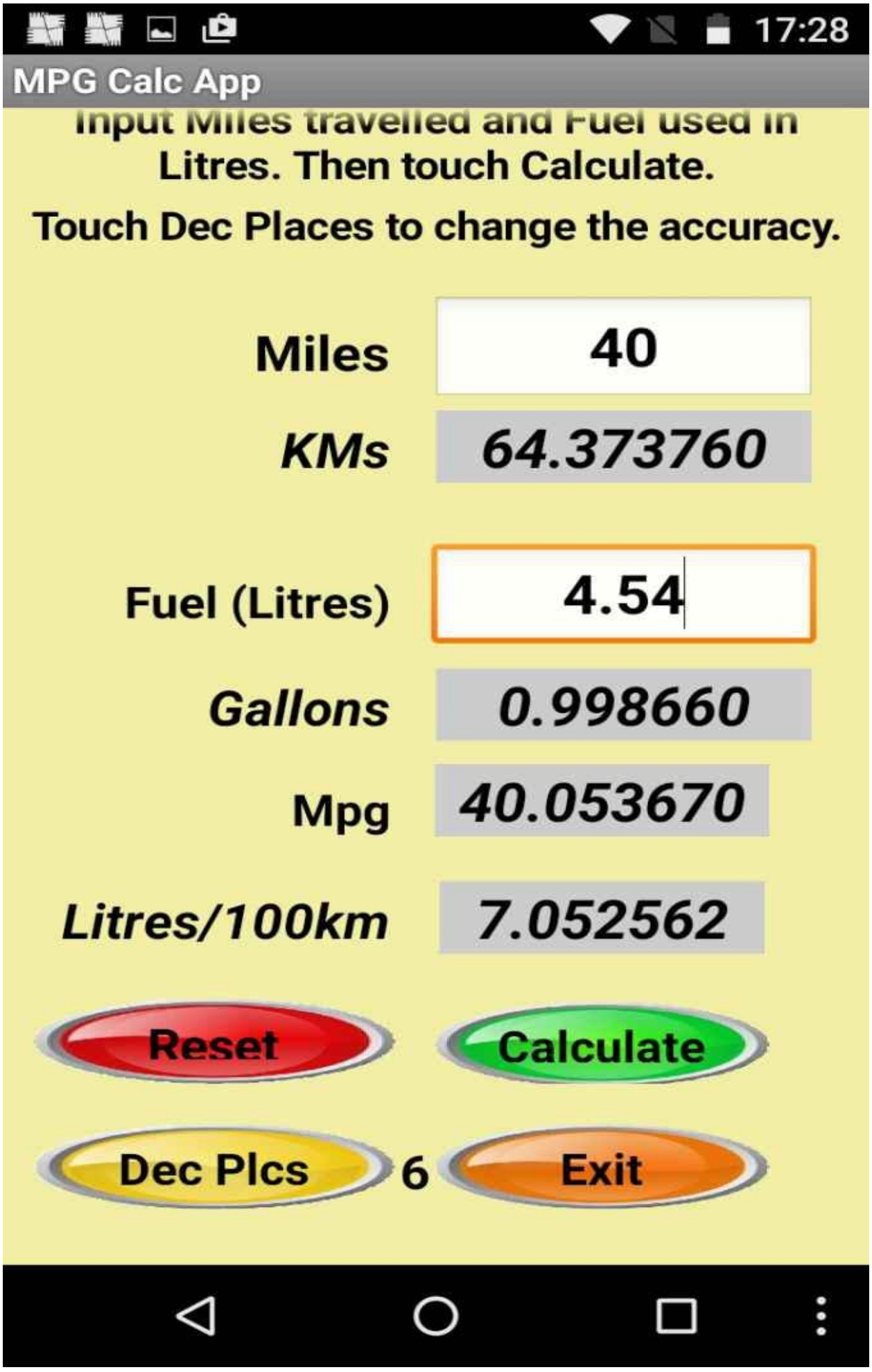
This time we start the same and work out the Gallons used(Var_Gallons).
Now we want to display the Gallons figure so we create Output_Gal using Var_Gallons constrained to the users Dec_Plc_Pick.
Next Output_KM, we multiply Miles_Input by 1.609344 constrained to the users Dec_Plc_Pick.
Next Var_MPG, we divide Miles_Input by Var_Gallons.
Next Output_Answer or MPG, Var_MPG to the users Dec_Plc_Pick.
And Finally OutputLitPer100, we divide 282.481 by Var_MPG to the users Dec_Plc_Pick.

The names aren't important, just so that they are unique and they make it obvious to you, or anyone you share the .aia file with, what they are.
This is what it looks like on a phone.


17:28
MPG Calc App
Input Miles travelled and Fuel used in Litres. Then touch Calculate.
Touch Dec Places to change the accuracy.
Miles
40
KMs
64.37
Fuel (Litres)
4.54
Gallons
1.00
Mpg
40.05
Litres/100km
7.05
Reset
Calculate
Dec Plcs
2
Exit

And below is what it looks like on a phone if we change the Decimal Places to 6.

Which is obviously more accurate than is required, but in some case you will need this range of accuracy.

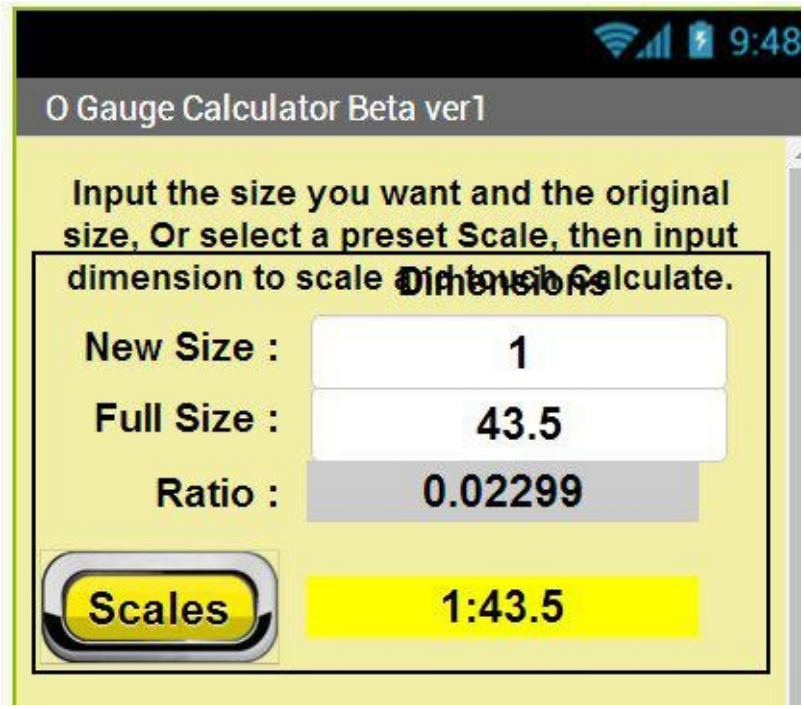

17:28
MPG Calc App
Input Miles travelled and Fuel used in
Litres. Then touch Calculate.
Touch Dec Places to change the accuracy.
Miles
40
KMs
64.373760
Fuel (Litres)
4.54
Gallons
0.998660
Mpg
40.053670
Litres/100km
7.052562
Reset
Calculate
Dec Plcs
6
Exit

Remember when adding new Labels, Text Boxes etc., to change the font from default, as well as sizes, bold, Italic or whatever else you want your standard to be.

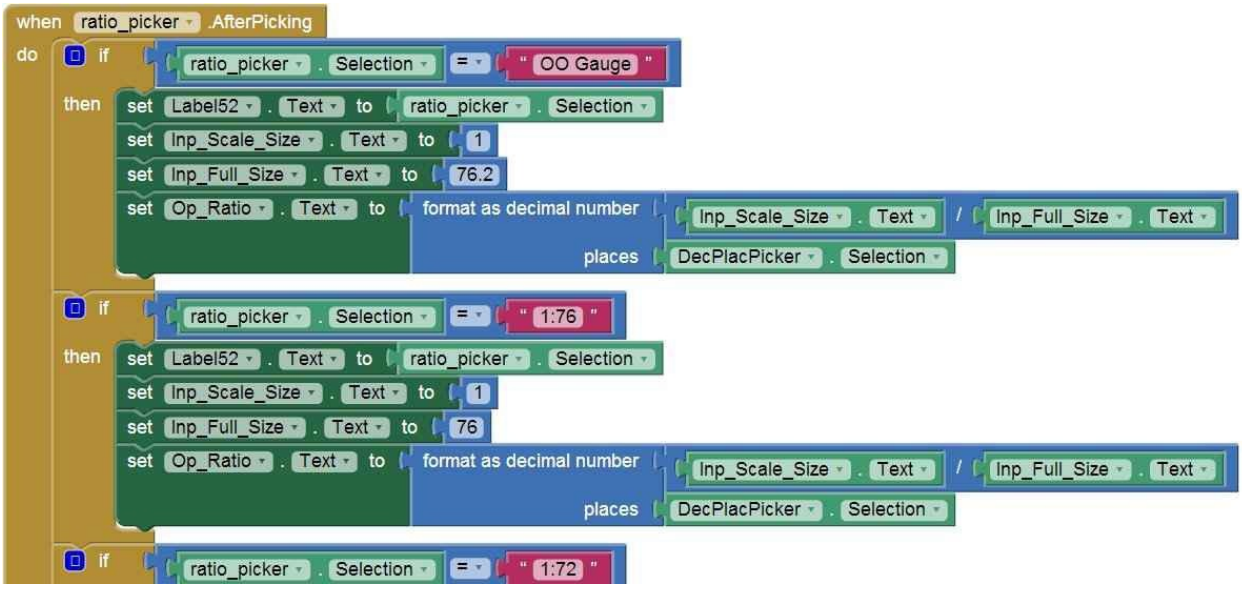
At this point I was going to start an exercise with 3 then 4 inputs. But chatting to friends once they got this far they said they were ready to do their own Apps and just needed some other help like if you want to give users a choice of 2 or 3 inputs how does it then work? So that's what I've done.

Choices

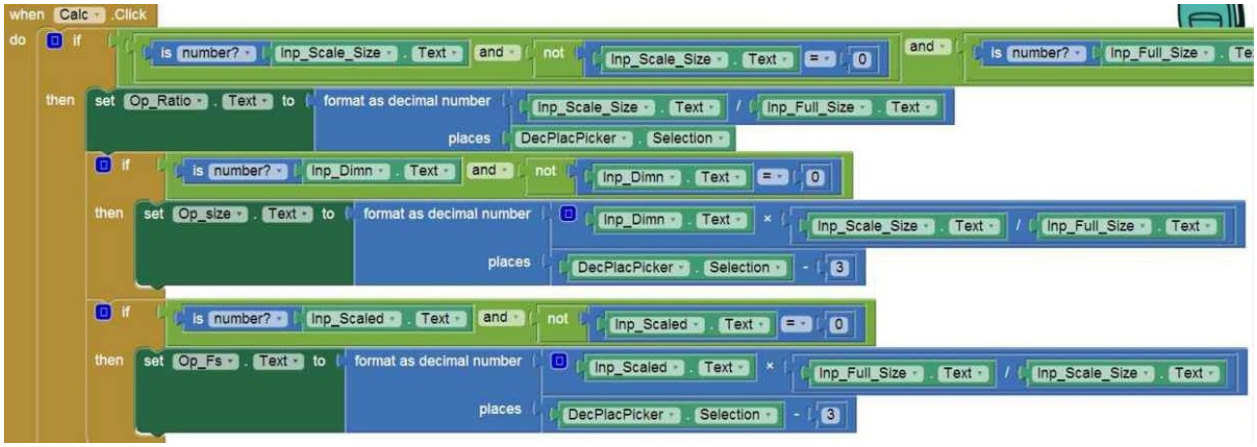
Well first of all we have various ways of giving User choices;
They can enter numbers using TextBoxes
They can select numbers using List Picker
Or both, a choice of a number of inputs or pick a set using ListPicker.



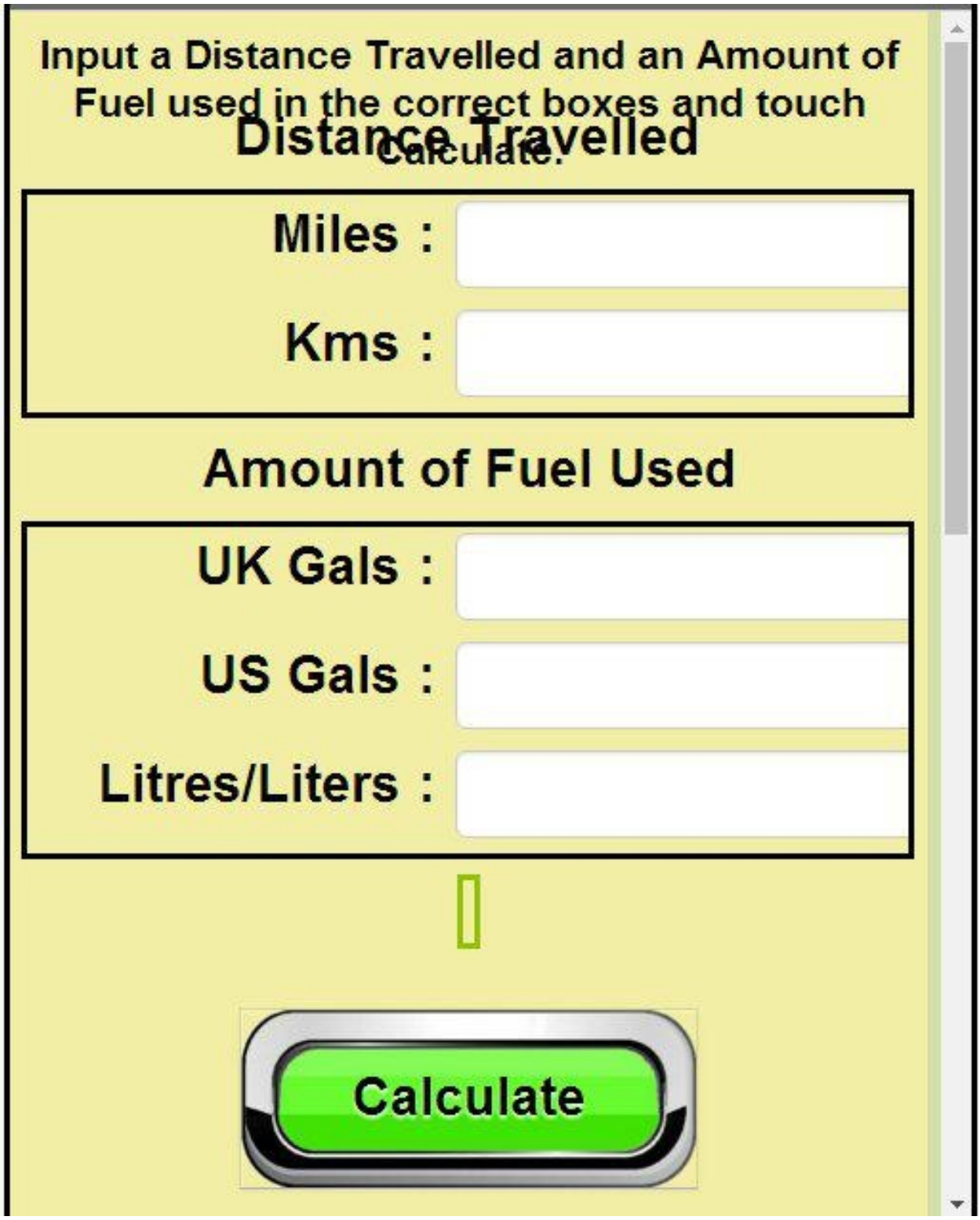
In the one above I've given the user both choices. They can put in 2 inputs or touch Scales and pick ones I've pre defined in ListPicker. In ListPicker I have put choices of "1:43, O Gauge, 7mm/1ft, 1:64, 1:72, 1:76, OO Gauge" each of these has pre set New size (Inp_Scale_Size) and Full Size (Inp_Full_Size) and will Calculate the Ratio (Op_Ratio), see below.



Because we set **Inp_Scale_Size** and **Input_Full_Size** the calculation doesn't need to be complicated it just checks, is there an **Inp_Scale_Size** and an **Input_Full_Size** value? Are they numbers but not Zero and if yes Calculate. See below.



Another way is to ask for say 2 Inputs but give a number of TextBoxes 3, 4, 5 or whatever and then create Blocks to check there are only 2 Inputs used and what to do with that particular 2.



In this one the user has the choice to Input a Distance travelled in Miles or KMs and the fuel used in UK Gallons, US Gallons or Litres. The Highlighted Box is my **Output_Solution.Text** Label. This gives the error messages and if all is correct will say that it used "Miles and Litres/Liters" or whichever were used as it's Inputs. (See screen view later)

So when Calculate is Touched we want 1 Distance and 1 Fuel Used Inputs.
So first we have;
If Miles is Empty **AND** Kms is Empty **AND** Uk Gals is Empty **AND** Us Gals is Empty **AND**

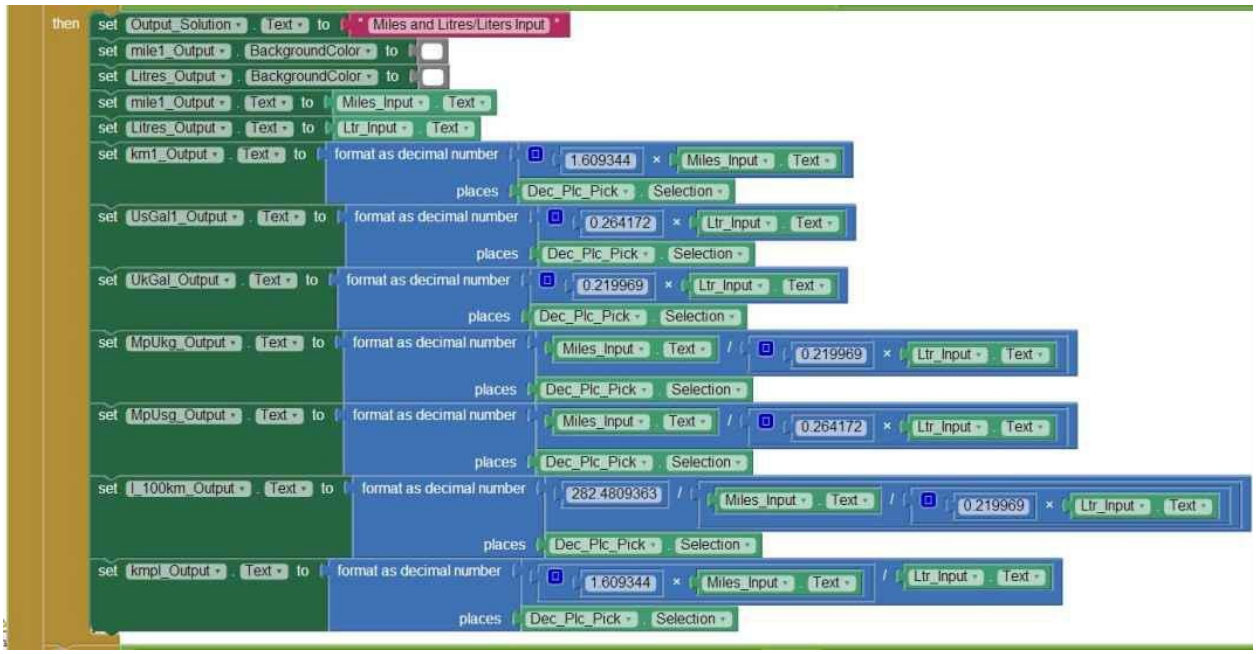
Litres is Empty, Solution = NEED 2 INPUTS!
We could check the Distance and Fuel Used separately so if the user Inputs only one and not the other we'd say we need one more input but it's more lines, if you want to, carry on it's your choice.

Now we need to check we only have one Distance
If miles is a number **AND** Kms is a number, Solution = ONLY ONE DISTANCE PLEASE!
The Same with the Fuel Used, only this is slightly more difficult
If UK Gals is a number **AND** US Gals is a number **OR** UK Gals is a number **AND** Litres is a number **OR** US Gal is a number **AND** Litres is a number, Solution = ONLY ONE FUEL USED PLEASE!

It's a bit long winded but can you see where we are going with this? We have to check there is only One Distance and only One Fuel Used to continue. If we had more choices we'd have more **AND** and **OR** bits.
For 2 choices you have One **AND**s and no **OR**s.
For 3 choices you have three **AND**s and two **OR**s.
For 4 choices you have six **AND**s and seven **OR**s.
And so on. We could run without this but then it will only display the last calculation blocks that were correct, as it would over write any previous correct ones and the user may not realise.

OK we have 2 Inputs but which one of six combinations? Hopefully you will realise that we now have to have 6 goes of working out the answers. I won't go through all the calculations as it would only work for this example so we can just go through one of the six and obviously the other five will all be similar but using different inputs to convert and then work out the same final solutions.

Suppose the user Inputs Miles and Litres used



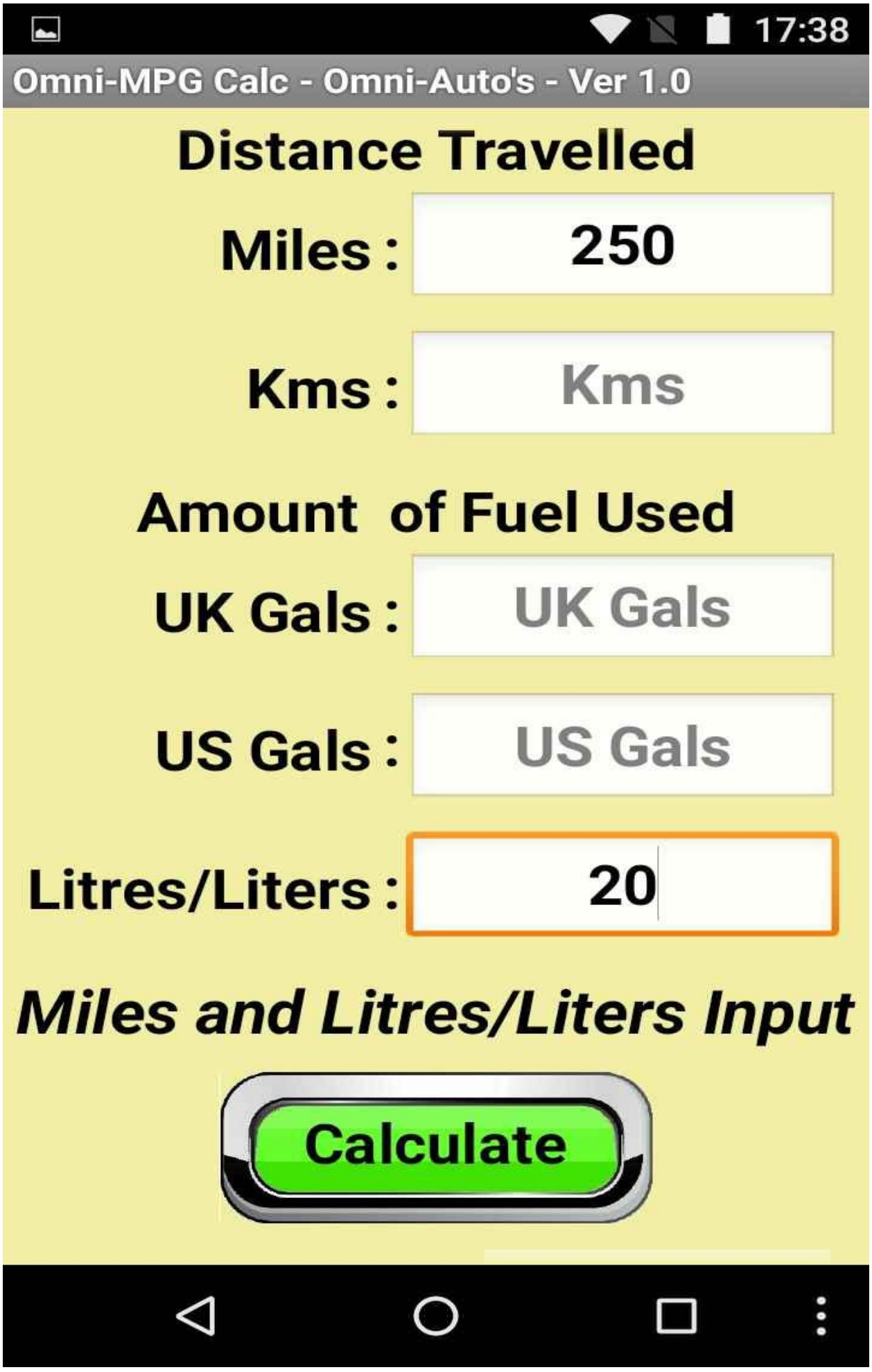
The **if** line, which was too long to display, says if Miles is a number and not Zero, and Litres/Liters Used is a number and not Zero and Kms is NOT a number and UkGal is NOT a number and UsGal is NOT a number then...

First **Output_Solution.Text** is set to Miles and Litres/Liters.
This tells you and the User what the Calculation has used as your inputs, you don't need this but it can be useful.
Now if you look at my Outputs, they have a light-grey background so one of the things I did was to change the background colour of the Outputs for (Miles and Litres/Liters) to White to make it clear that they were used as Inputs when just looking at the Output, as on a phone this is all you see. (See Output Screen later).
Then Display our Inputs Miles and Litres.
Then Convert Miles to KMs to the users Decimal Place Choice.
Then Convert Litres to UK Gals to the users Decimal Place Choice.
Then Convert Litres to US Gals to the users Decimal Place Choice.
Then Work out the Mile/Uk Gal to the users Decimal Place Choice.
Then Work out the Mile/Us Gal to the users Decimal Place Choice.
Then Work out the Litres per 100KM to the users Decimal Place Choice.
Then Work out the KM/Litres to the users Decimal Place Choice.

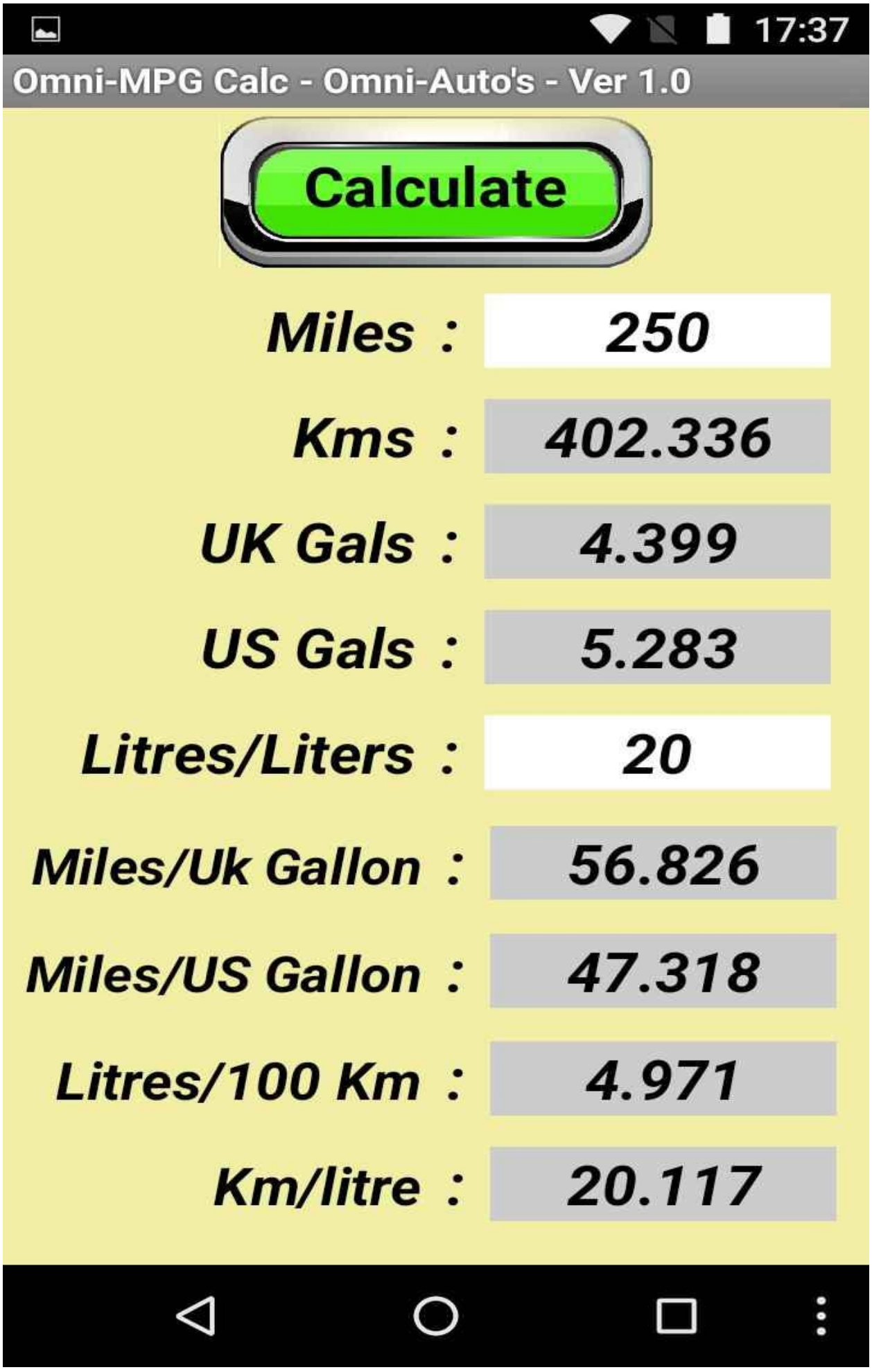
And now we do a similar routine for Miles and UK gallons, Miles and US Gallons, KMs and Litres, KMs and UK gallons and finally KMs and US Gallons.

Because we are changing BackGroundColor, each time the Calculate_But is run we must reset the BackGroundColor to "light-grey" again, we don't know which has changed so we reset all the outputs. And we reset all output text to "**get start value**". This is something else you don't need to do, but if you don't and the inputs are invalid and you don't notice the error message you would still see the last set of results in the Outputs!

This is the final look of the App on a phone, below is the Input area.


17:38
Omni-MPG Calc - Omni-Auto's - Ver 1.0
Distance Travelled
Miles :
250
Kms :
Kms
Amount of Fuel Used
UK Gals :
UK Gals
US Gals :
US Gals
Litres/Liters :
20
Miles and Litres/Liters Input
Calculate

And below is the Output/results area.


17:37
Omni-MPG Calc - Omni-Auto's - Ver 1.0
Calculate
Miles : 250
Kms : 402.336
UK Gals : 4.399
US Gals : 5.283
Litres/Liters : 20
Miles/Uk Gallon : 56.826
Miles/US Gallon : 47.318
Litres/100 Km : 4.971
Km/litre : 20.117

Hopefully you noticed and realised that I didn't use variables in this one and duplicated the equations using the inputs to avoid errors. Be careful if you use the same equation twice, if you modify them one at any time, don't modify each copy, actually **Duplicate** the equation block you are less likely to make mistakes. There is no wrong way here so long as all your answers come out correct. This was an early App before I'd started using Variables, the important thing to remember is once you've rounded up a number(formatted it to a set number of Decimal Places) don't use it for an equations, it's only good for an Display.

Odds and sods and some reminders

Copy and paste

When editing blocks you can **Duplicate** the block or you can highlight the text/numbers inside it and copy(delete, paste, etc.) them using the right mouse button. But in the Designer window those mouse functions are turned off. If you're old enough you probably remember this but you used to select and copy and paste etc. using the keyboard. Well you still can.

Text selecting.
Get the cursor over the start or end of text and then keeping your finger on the Left Mouse Button (Select) drag it until you have highlighted all you want. You can also do this using the shift + Cursor keys but as the mouse will do this, it's easier with the mouse.

Text Copying.
While it's highlighted Press **Ctrl** and **C**, this copies it to your clipboard.

Text Pasting.
Now find where you are putting it, click the cursor there using the mouse.
And press **Ctrl** and **V**
Or press **Shift** and **Insert**

If you make a mistake whilst you're in a text box you can scroll back through your entries by pressing **Ctrl** and **Z**
You can scroll forward again through your entries by pressing **Ctrl** and **Y**
As soon as you leave that text box the clipboard clears so you can only scroll through once you've done something in that box, whilst you're still in it.

Order

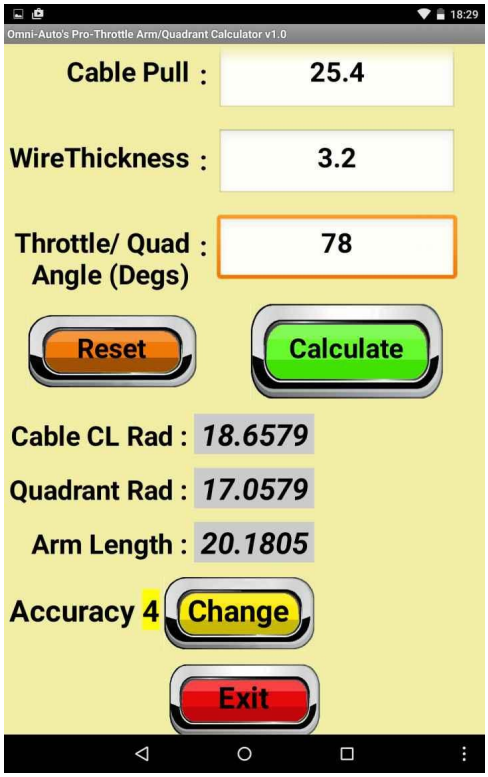
The order of your lines of blocks are important. If like me you have a error message output line on your display, if you have a number of errors only the last error output will be displayed. So put the error checking parts in order of most useful last as this will over write any previous ones.

It should go without saying this is basically a series of commands so it work through them one at a time from the top down so you may have all the right lines but if one is out of order it might fail.

Beta Testing

Get as many people as you can to beta test your App or play with it and ask them not to spare your feelings get them to say what they really think. Careful here if your writing something that works out say bend deductions for aluminium only give it to somebody who has used bend deductions and knows what they are and what they are for to test the usability of it. By all means let anybody play with it to see if they can cause an error. For instance with the bend deduction one, someone might put an angle of 360° which would crash the program. Anyone who knew what they were doing would never do that and you'd never know that you hadn't done an error checking line to stop angles outside the range it will work for. This can be a problem if you are working on something you know too much about, you'll only input values that you know will work, after all who'd put in a value that caused it to crash? Probably the first person to try it after you said it was finished. With my suspension program I have to put in checks that upper parts are above lower parts and that outboard parts are outboard of the inboard parts. It's obvious but if someone accidentally puts in the wrong number it could either crash the App or worse come out with results that may appear to be great but are in fact rubbish. Holding my hands up I remember saying years ago "but why would anyone enter numbers like that!" When I was faced with a user who'd crashed my all singing, all dancing program, it only worked if you knew what to input. They need to be foolproof and I guess this is even more true for ones you give away. People will download it and try it when it's free, they may have no idea about how to use your App but if it fails/returns an error or in anyway appears not to work you can bet they will let everyone know.
The more you create and use the easier and hopefully the better you will become at creating Apps.

App Themes



All my Apps use a common theme :-

A pale yellow background.
Incorporated Status bar across the top.
White boxes for User inputs.
Light-Grey boxes for calculated outputs.
Yellow boxes for changeable numbers as in number of decimal places, this can be changed.
Green Buttons will calculate the Inputs.
Yellow Buttons this is something that can be changed and then automatically recalculated.
Amber Button is the reset button which returns all values to their start value.
And the Red Button is to Exit or Quit the App, and as with all Apps you should then go to the "Recent Apps" or "Multitasking View" and close them properly after.

This may not work for you but if you are going to make a series of Apps try to be consistent across them so users feel at home when they open them.

Starter

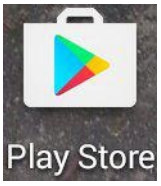
At some point create a **Starter** project. Don't do it at the start you won't know what you need. Once you have created two or three of your own Apps you might have some idea but certainly when you've come up with a look for your Apps save the latest, as Project "**Starter**". You can change the AppName to unknown or TBA whatever and change the VersionCode to 10 and VesionName to 1.0, and AboutScreen to ** - This App will work out **. It's up to you if you want to take out the equations or leave them in case you want to modify them. But save it and then, when you start your next project open "**Starter**" and do a "save project as" and give the new name and start editing and then create your new App.
You may want to change your **Starter** Project from time to time as you find more things you want to include in your Apps, you may also want to revisit earlier Apps and update them so they still fit in with your look/theme call it what you will.

Updates

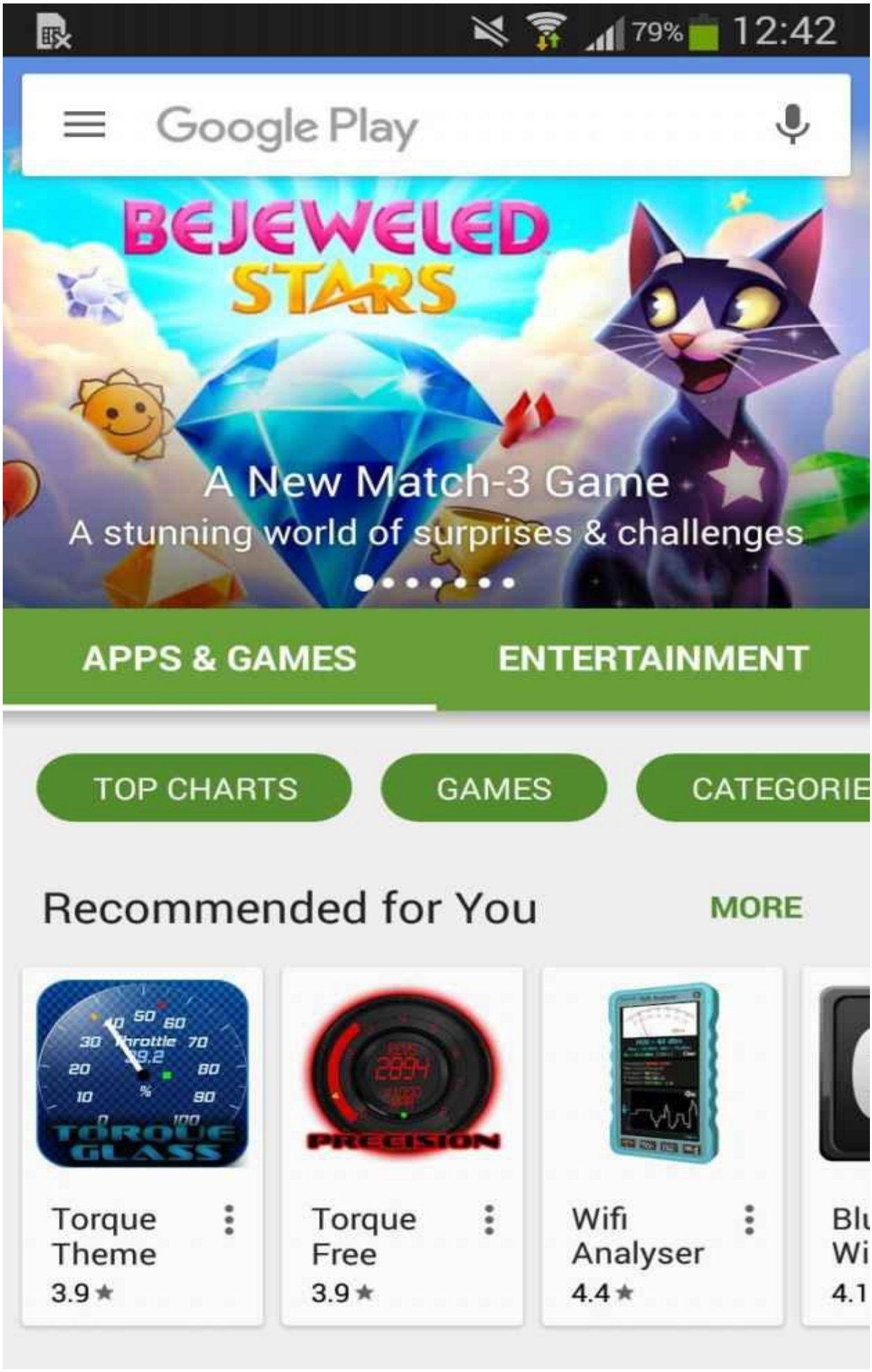
You may want to update Apps. First thing is to open the project you are updating and "save as" and use the old project name followed by the VersionName. I.e. MPG at version 1.0 would be saved as MPG_10. Don't change the AppName. As soon as that's done go to Project, there should be MPG and MPG_10 find your original one (MPG) and do any changes in that, start with VersionName and Version Code. I start with **1.0** and **10** and then, **1.1** and **11** if it's a small change, or **2.0** and **20** if it's major. And so on.

Why are we doing this?
First you have a copy at each release or version, so we can go back to if you get any problems further down the line.
And second if you sell or want update the App on your Device then it must be the same AppName and a higher Version to update.
It might be wise to do this as soon as you release a Version to anyone, so that you have a record of what has gone out and in what state. Or else you could let one friend have copy and then you change something and give a copy to someone else and you carry on working on it. You now have three versions of it and they all say they are the same version but might have different faults so any feed back might be a waste of time.

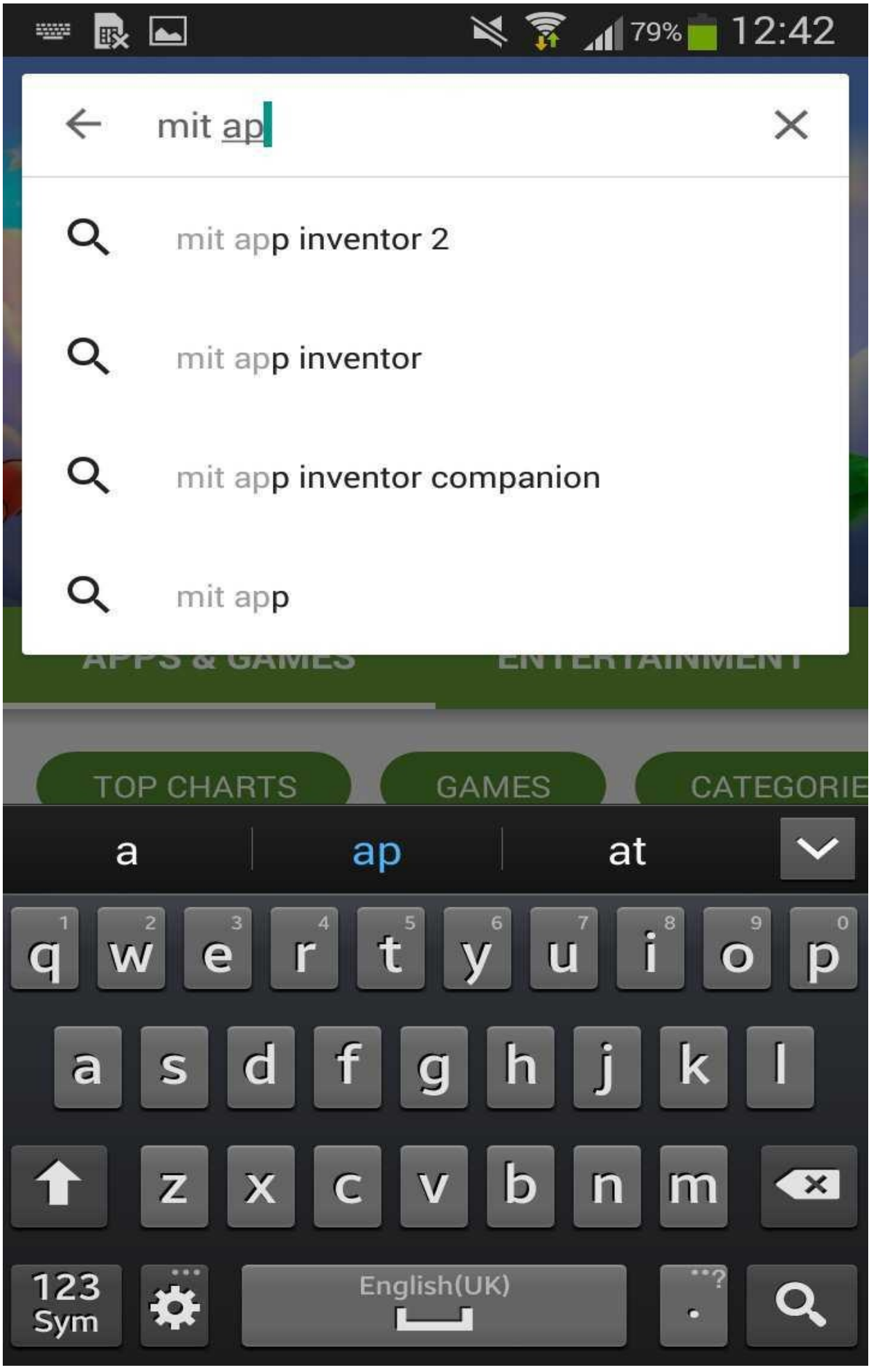
Installing The MIT AI2 COMPANION App

Go to **Google Play Store** 

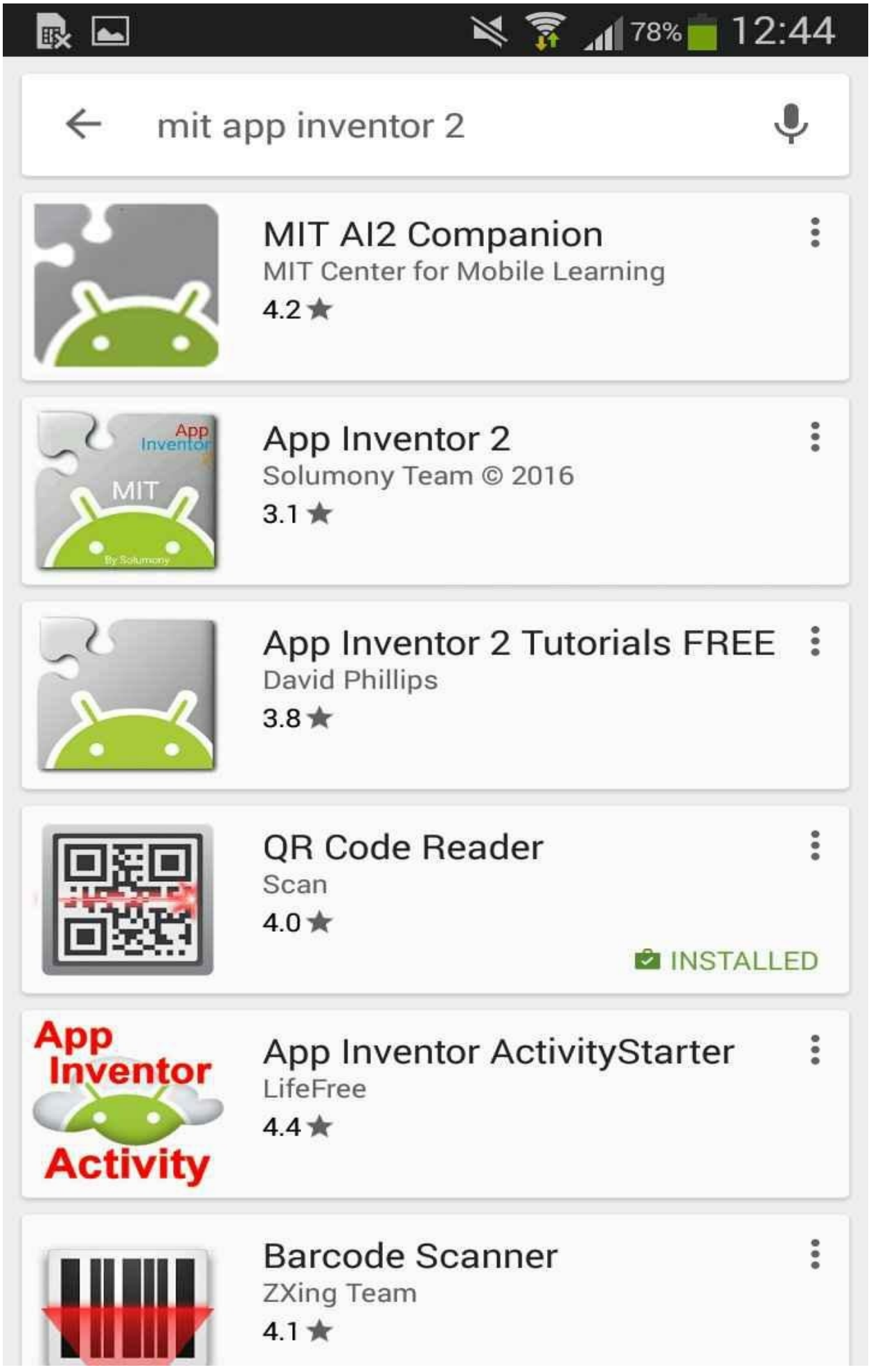
(making sure your wireless is on) if you use Android


79%
12:42
Google Play
BEJEWELED
STARS
A New Match-3 Game
A stunning world of surprises & challenges
APPS & GAMES
ENTERTAINMENT
TOP CHARTS
GAMES
CATEGORIE
Recommended for You
MORE
Torque
Theme
3.9
Torque
Free
3.9
Wifi
Analyser
4.4
4.1

And type or say **MIT AP** in the search area.


79%
12:42
mit ap
mit app inventor 2
mit app inventor
mit app inventor companion
mit app
TOP CHARTS
GAMES
a
ap
at
English(UK)
123
Sym

You should see **mit app inventor 2** select it or **mit app inventor companion**


78%
12:44
mit app inventor 2
MIT AI2 Companion
MIT Center for Mobile Learning
4.2
App Inventor 2
Solumony Team © 2016
3.1
App Inventor 2 Tutorials FREE
David Phillips
3.8
QR Code Reader
Scan
4.0
INSTALLED
App Inventor ActivityStarter
LifeFree
4.4
Barcode Scanner
ZXing Team
4.1

You should see "**MIT AI2 COMPANION** by MIT Center for Mobile Learning". Select this one. Note you may also notice **QR Code Reader** by Scan, at some point you may also want this as it will speed up downloads and connections for **MIT App Inventor**.

MIT AI2 Companion

MIT Center for Mobile Learning

3 PEGI 3

INSTALL

500 THOUSAND
Downloads

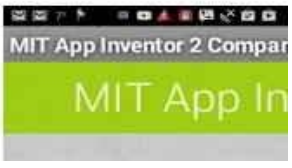
4.2 ★★★★½
13,774

Education

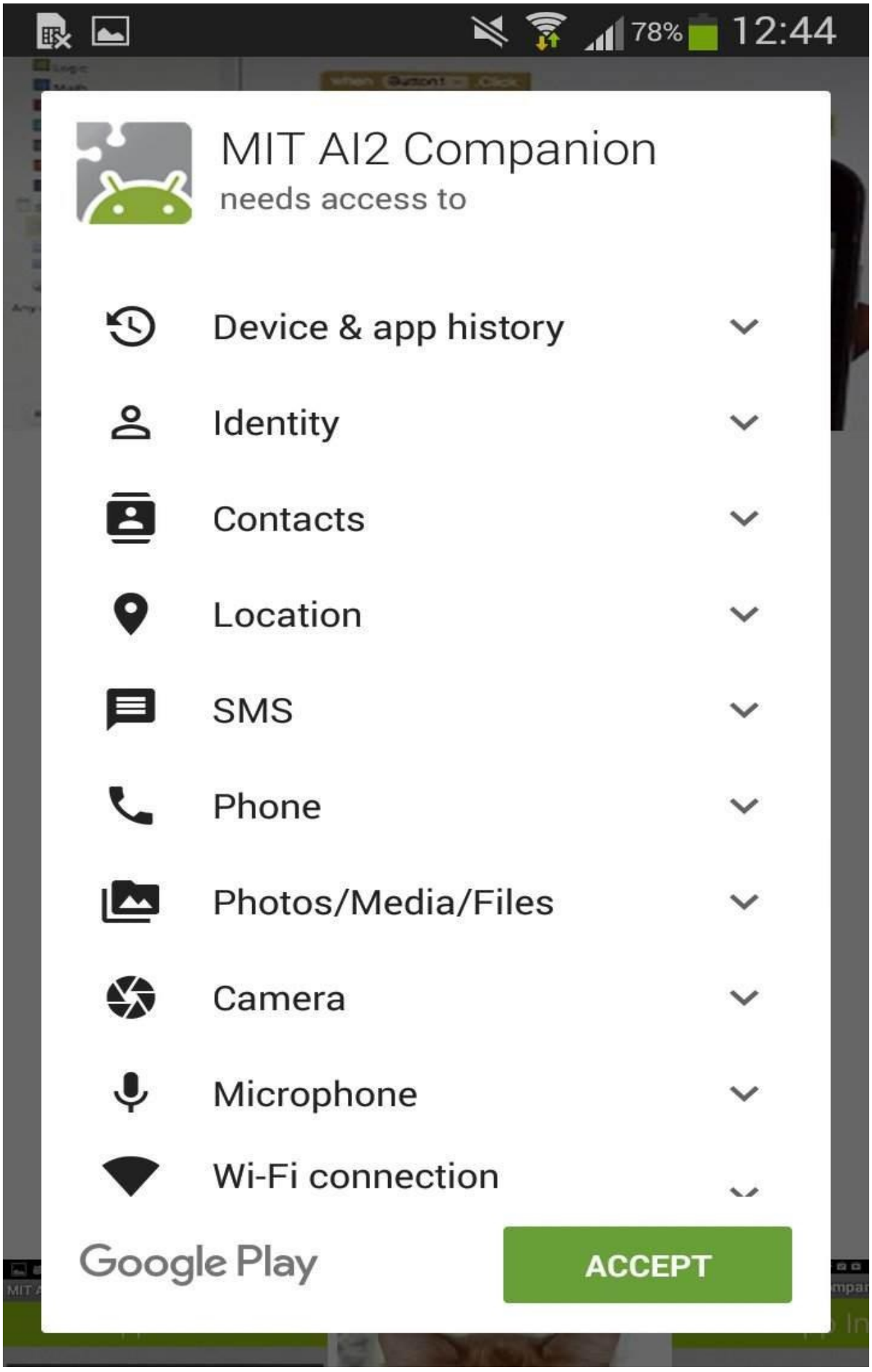
Similar

Develop your very own Android Applications using MIT App Inventor 2!

READ MORE



Then select **INSTALL**


78%
12:44
MIT AI2 Companion
needs access to
Device & app history
Identity
Contacts
Location
SMS
Phone
Photos/Media/Files
Camera
Microphone
Wi-Fi connection
Google Play
ACCEPT

It will then ask for Access to (see above) you have to decide if you want to use the App then touch **ACCEPT**. If not you'll have to use the Android Emulator, on your Computer, it works but it's not anything like using it on a **Device**.

MIT AI2 Companion

MIT Center for Mobile Learning

3 PEGI 3

UNINSTALL OPEN

500 THOUSAND Downloads

4.2 ★★★★½ 13,774

Education

Similar

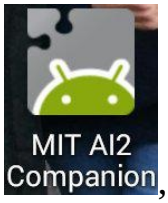
Develop your very own Android Applications using MIT App Inventor 2!

WHAT'S NEW

Updates to MIT App Inventor version nb147

That's it installed, if you are just starting you don't need this yet, if you have created your first design for your App then **OPEN**.

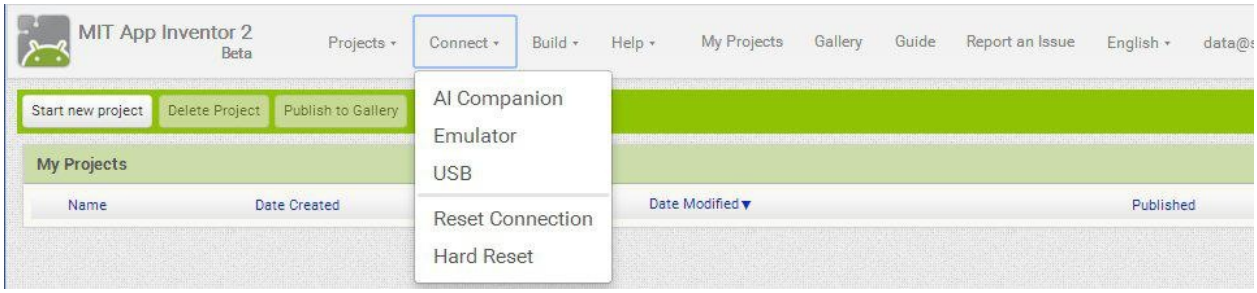
Using The App

Ok you've either **OPEN**need the App on installation or selected the 

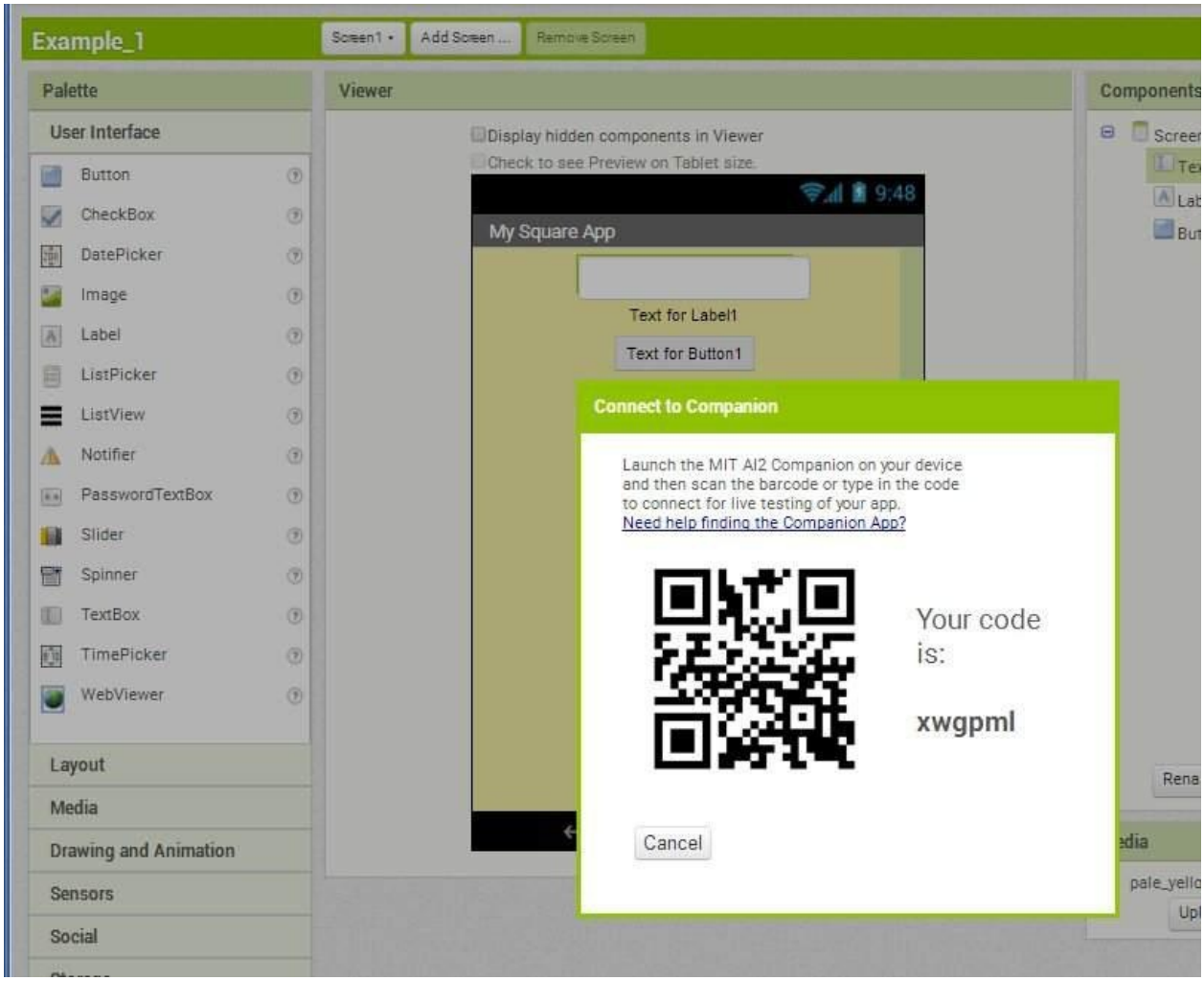
, **MIT AI2 COMPANION** Icon.


78%
12:46
MIT App Inventor 2 Companion
MIT App Inventor 2
type in the 6-character code
-or-
scan the QR code
Six Character Code
connect with code
scan QR code
Your IP Address is:
Version: 2.36

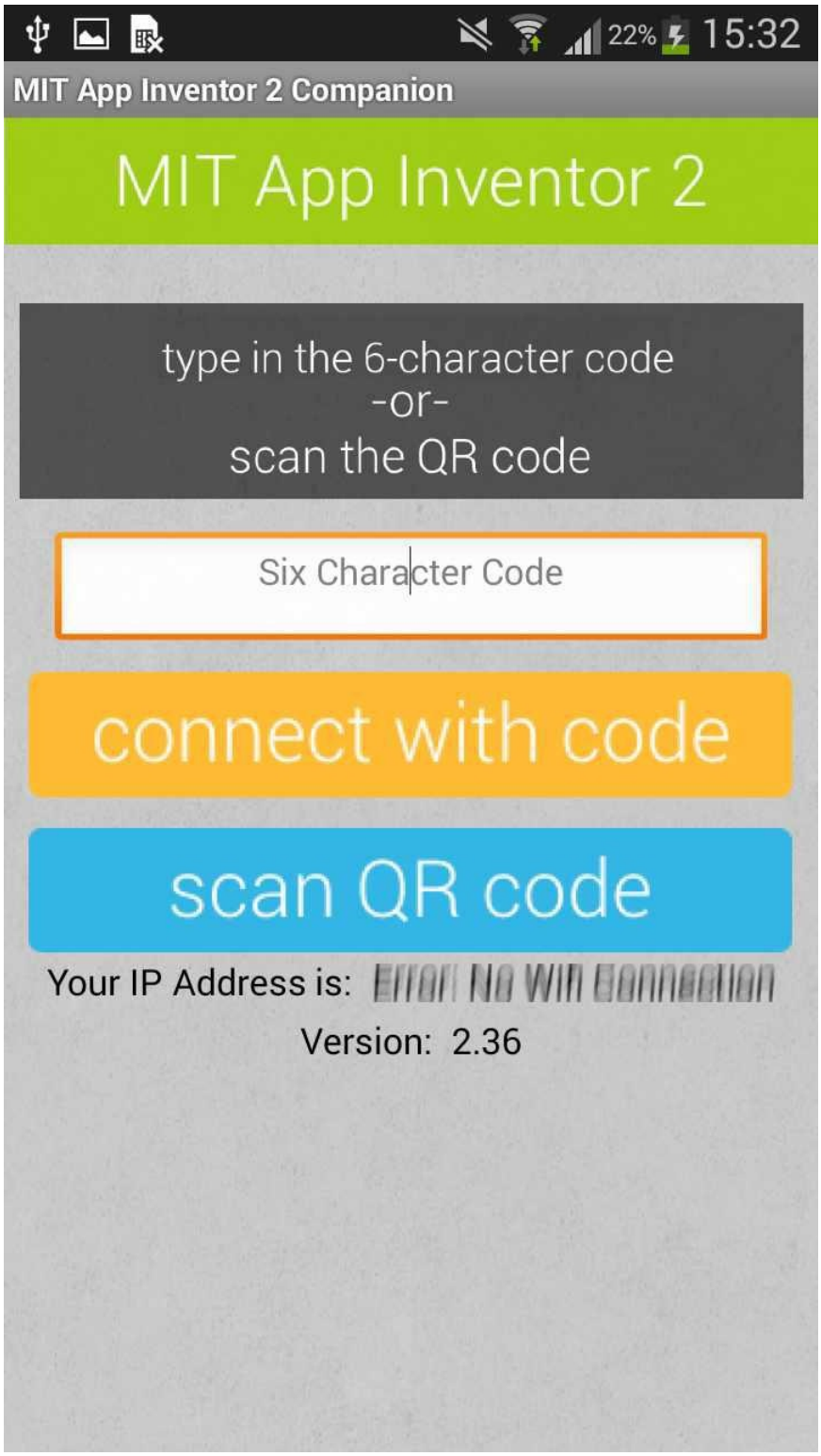
We'll assume your on the **MIT App Inventor2** page (http://ai2.appinventor.mit.edu/) on your Computer browser and that you have selected **Connect** the command row at the top.



You must have a project created and open before you can do this.
Click "**A1 Companion**" and this will appear.



The Code will change every time you use it. You now have a choice, on your device you can input (in this case) xwgpml and touch "connect with code".



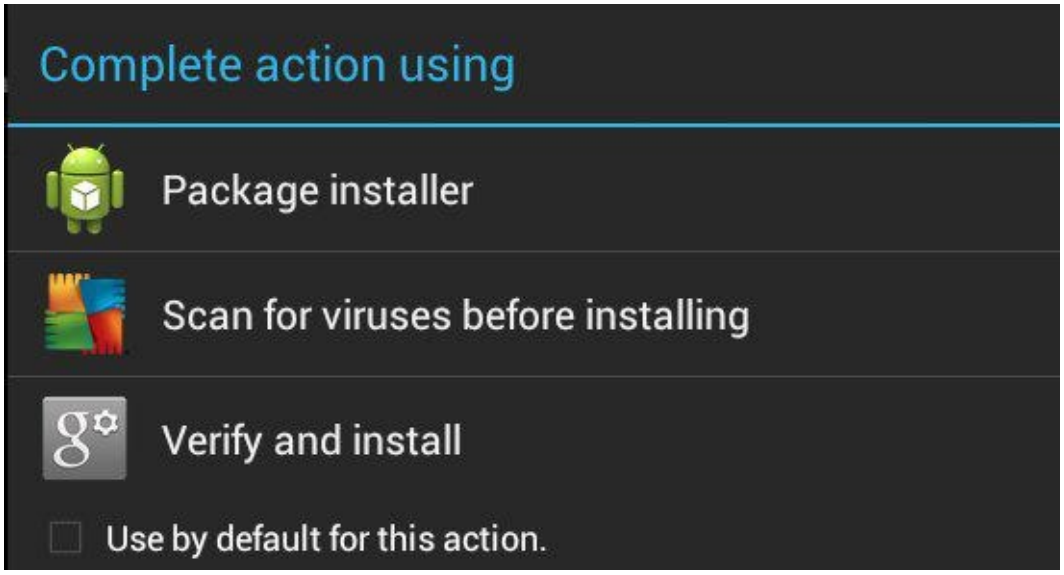
Or choose "Scan QR Code" and scan it then touch "connect with code" when it's finished. Yes the QR code option types the code in for you nothing more than that.

You are now connected and as you, change any properties, or add or delete block the Device should update.

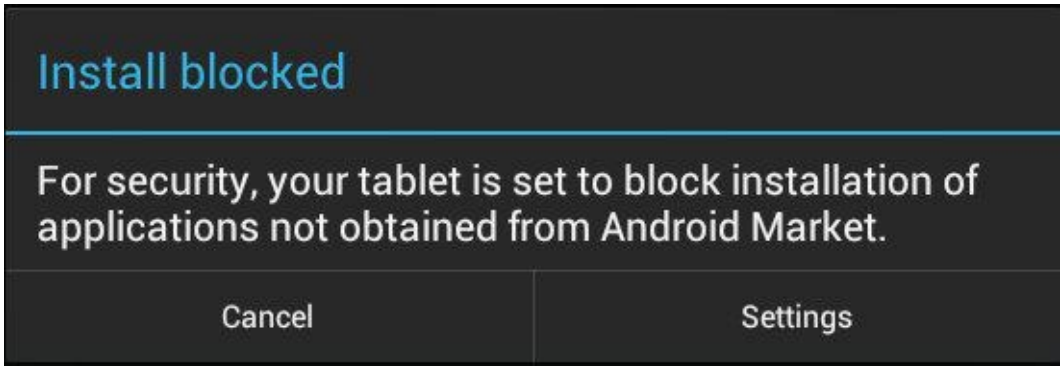
Installing Apps manually

You can install Apps manually following these steps:

Download the installation file to your computer or Tablet.
Make sure the file is in **.apk** format which is supported by the Android Tablet system.
If its on your Computer, connect the Computer to your Tablet, with a USB cable.
Copy the **.apk** file to your Tablet.
Disconnect your Tablet form the computer.
Turn on your Tablet and select File Explorer.
Find the **.apk** file and double tap it.



You will be given some choices, select one and you will probably get the following message appear.



Select **Settings** then **Security - Device Administration - Unknown sources** tick the box, you can go back and untick it, after your App has loaded. If you leave it ticked it will not warn you if you are about to install an App from an Unknown source.
Follow the on-screen instructions to install the App.

Once the App is installed successfully, you can Open it and try it or Go to your Home Screen find where you want to put and find it in the App tab and place on your screen.

Or Use the Play Store.

Uninstall Apps

Display all Apps by touching the App Tab.
Select **Menu**, **Manage Apps**and touch on the App that you want to Uninstall.
Touch on **Uninstall** to Uninstall the App.
Or
Go to **Settings** then **Apps** then scroll down and find the App you don't want, select it and touch **Uninstall**.

What Now

Well you have created your App and tested it and hopefully let others try it out and then looked at any feedback and address any problems, what now. Well I guess you could put it on your Device and use it or put it out there for others. You can sell or give it away the steps will be the same. You need to register as an App developer with Amazon and or Google. I choose Amazon as it's free, and most of mine I will give away, Google charge $25 per year. Maybe if I sell loads I would consider putting my Apps on Google but as I will be giving most away and the ones I sell will be for a minimal amount and fairly niche, I doubt I will bother.
Once you sign up you can start filling out 6-7 pages of data about each App as you add them.
The first is the Name of your App.

Terms of Service

To use App Inventor for Android, you must accept the following terms of service.

Terms of Service

MIT App Inventor Privacy Policy and Terms of Use

MIT Center for Mobile Learning

Welcome to MIT's Center for Mobile Learning's App Inventor website (the "Site"). The Site runs on Google's App Engine service. You must read and agree to these Terms of Service and Privacy Policy (collectively, the "Terms") prior to using any portion of this Site. These Terms are an agreement between you and the Massachusetts Institute of Technology. If you do not understand or do not agree to be bound by these Terms, please immediately exit this Site.

MIT reserves the right to modify these Terms at any time and will publish notice of any such modifications online on this page for a reasonable period of time following such modifications, and by changing the effective date of these Terms. By continuing to access the Site after notice of such changes have been posted, you signify your agreement to be bound by them. Be sure to return to this page periodically to ensure familiarity with the most current version of these Terms.

Description of MIT App Inventor

From this Site you can access MIT App Inventor, which lets you develop applications for Android devices using a web browser and either a connected phone or emulator. You can also use the Site to store your work and keep track of your projects. App Inventor was originally developed by Google. The Site also includes documentation and educational content, and this is being licensed to you under the Creative Commons Attribution 4.0 International license (CC BY 4.0).

Account Required for Use of MIT App Inventor

In order to log in to MIT App Inventor, you need to use a Google account. Your use of that account is subject to Google's Terms of Service for accounts, and the information you provide to Google is governed by Google's Privacy Policy. MIT has no access to your Google account or the information you provide for it other than the account email address, which we may use to contact you. In the future, the Center for Mobile Learning may provide alternative means for using MIT App Inventor without Google accounts, at which point we will update these terms of use to reflect those alternatives.

Information about you

Beyond the name of your Google account, you do not have to tell us anything about yourself to download the App Inventor setup software or use the MIT App Inventor Web site. From time to time, we will give you the option of telling us some things about yourself, but you do not have provide this information if you don't want to. Please understand that by submitting any Personally Identifiable Information to us, you consent and agree that we may collect, use and disclose such Personally Identifiable Information in accordance with these Terms, and as permitted or required by law. If you do not agree with these Terms, then please do not provide any Personally Identifiable Information to us.

We track information indicating, among other things, which pages of our Site were visited, the order in which they were visited, when they were visited, and which hyperlinks and other user interface controls were used. We also track information about user projects and how those projects are developed when connected to Android devices.

We may log the IP address, operating system and browser software used by each user of the Site, and we may be able to determine from an IP address a user's Internet Service Provider and the geographic location of his or her point of connectivity. Various web analysis tools are used to collect this information. Some of the information is collected through cookies (a small text file placed on your computer). You should be able to control how and whether cookies will be accepted by your web browser. Most browsers offer instructions on how to reset the browser to reject cookies in the "Help" section of the toolbar.

Among other things, we may use the information that you provide (including your Personally Identifiable Information) in connection with the following

In order to debug and improve the MIT App Inventor system.
For purposes of scientific research, particularly, for example, in the areas of how people learn and create with MIT App Inventor.
For the purpose for which you specifically provided the personal information, for example to respond to a specific inquiry or provide you the specific course and/or services you select.
To publish information gathered about learning and creating with MIT App Inventor but only as non-personally identifiable data.
As otherwise described to you at the point of collection.
Sharing with Third Parties

We may share the information we collect with third parties as follows:

With service providers or contractors that perform certain functions on our behalf, including processing information that you provide to us on the Site or operating the Site or portions of it. These service providers and contractors will be obligated to keep your information confidential.
With research collaborators, but only under the condition that they are obligated to keep any personally identifying information confidential.
To respond to subpoenas, court orders, or other legal process, in response to a request for

cooperation from law enforcement or another government agency, to investigate, prevent, or take action regarding illegal activities, suspected fraud, or to enforce our user agreement or privacy policy, or to protect our rights or the rights of others.

The Apps you create

By creating and storing apps on the MIT App Inventor server you represent and warrant that you are the owner and creator of the apps, (i) that you have the authority to authorize MIT to store the apps on the MIT App Inventor Server and (ii) you will use the apps in compliance with all applicable laws and regulations. You, and not MIT, are solely responsible for your apps and your use of them.

MIT has no proprietary rights in the apps you create with MIT App Inventor. These apps belong to you. Your apps are stored on the MIT App Inventor server. You have the right to download your apps and delete them from the server at any time. If you delete an app, there is the possibility that MIT may be able to continue to access it from the backups we keep for purposes of system maintenance, but these backups are periodically purged and are not designed for long-term preservation. MIT will strive to keep your apps and your account accessible to you for as long as you wish, but we have no obligation to do so, and MIT has no liability for the consequences of the service becoming unavailable or your apps becoming unavailable. We therefore strongly suggest that you maintain backup copies of valuable apps at places besides the MIT App Inventor server.

User Postings

MIT App Inventor has a gallery for sharing apps, user profile information and commentary. You agree that you are responsible for your own use of the Site and your User Postings. **User Postings** include all projects, applications and content submitted, posted, published or distributed on the Site by you or other users of the Site.

By submitting or distributing your User Postings, you affirm, represent, and warrant that you are the creator and owner of or have the necessary licenses, rights, consents, and permissions to reproduce and publish the User Postings, and to authorize MIT and the Site's users to reproduce, modify, publish, and otherwise distribute your User Postings as necessary to exercise the license granted by you below. You, and not MIT, are solely responsible for your User Postings and the consequences of posting or publishing them.

All content and projects on this Site are licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. Therefore, by submitting any User Posting you hereby license it under the CC SA license, which permits anyone to view, modify, and redistribute these materials.

MIT has no proprietary rights in your User Postings. You have the right to download or delete them from the server at any time.

Community Guidelines

You are required to follow these community guidelines.

Part of being a good member of this Site's community means that you agree that you will use the Site in compliance with these Terms of Service and all applicable local, state, national, and international laws, rules and regulations, including copyright laws and any laws regarding the transmission of technical data exported from your country of residence and all United States export control laws. You understand that certain types of content and certain destructive actions listed below are strictly prohibited on the Site and you agree that you will not post any content or partake in any activity listed.

Being a good member of this Site's community also means helping to keep App Inventor a friendly and creative community - where people from all around the world feel welcome.

Be respectful. When sharing projects or posting comments, remember that people of many different ages, and from different places and cultures, will see what you've shared.

Be constructive. When posting or commenting on other's work, say something you like about it and offer constructive suggestions.

Share. You are free to remix projects, ideas, images, or anything else you find in the App Inventor gallery and anyone can use anything that you share. Be sure to give credit when you remix.

Keep personal info private. Your projects on the Site are private unless you share them. But all other information is public. Many people choose to guard their privacy by never using their real names in posts, and never posting contact information like phone numbers or addresses, and we encourage you to do likewise. Young people, especially, should be careful about placing personal information on the Site.

Help keep the Site friendly. If you think a project, comment or user profile is mean, insulting, too violent, or otherwise inappropriate, click **Report** to let the moderators know about it.

Prohibited Content

Content that defames, harasses, or threatens others
Content that discusses illegal activities with the intent to commit them
Content that infringes another's intellectual property, including, but not limited to, copyrights, or trademarks
Any inappropriate, profane, pornographic, obscene, indecent, or unlawful content
Advertising or any form of commercial solicitation
Political content or content related to partisan political activities
Content that contains intentional inaccurate information with the intent of misleading others.
Prohibited Activities

Viruses, trojan horses, worms, time bombs, corrupted files, malware, spyware, or any other similar software that may damage the operation of another's computer or property
Using the Site in any manner intended to damage, disable, overburden, or impair any MIT server, or the network(s) connected to any MIT server, or interfere with any other party's use and enjoyment of the Site.
Attempting to gain unauthorized access to the Site, other accounts, computer systems or networks connected to any MIT server through hacking, password mining or any other means.
Obtaining or attempting to obtain any materials or information stored on the Site, its servers, or associated computers through any means not intentionally made available through the Site.
Use of MIT Names and Trademarks

"MIT App Inventor", "MIT", "Massachusetts Institute of Technology", and its logos and seal are trademarks of the Massachusetts Institute of Technology. You may not use MIT's names or logos, or any variations thereof, without prior written consent of MIT. You may not use the MIT name in any of its forms nor MIT seals or logos for promotional purposes, or in any way that deliberately or inadvertently claims, suggests, or in MIT's sole judgment gives the appearance or impression of a relationship with or endorsement by MIT.

The Digital Millennium Copyright Act ("DMCA")

It is MIT's policy to respond to notices of alleged copyright infringement that comply with the Digital Millennium Copyright Act. Copyright owners who believe their material has been infringed on the Site should contact MIT's designated copyright agent at dmca-agent@mit.edu, or at 77 Massachusetts Ave., Cambridge, MA 02138-4307 Attention: MIT DMCA Agent, W92-263A.

Notification must include:

Identification of the copyrighted work, or, in the case of multiple works at the same location, a representative list of such works at that site.
Identification of the material that is claimed to be infringing or to be the subject of infringing activity. You must include sufficient information for us to locate the material (e.g., URL, IP address, computer name).
Information for us to be able to contact the complaining party (e.g., email address, phone number).
A statement that the complaining party believes that the use of the material has not been authorized by the copyright owner or an authorized agent.
A statement that the information in the notification is accurate and that the complaining party is authorized to act on behalf of the copyright owner.
Disclaimer of Warranty / Indemnification/Limitation of Liabilities

THE SITE IS PROVIDED "AS IS" AND "AS AVAILABLE" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR USE FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. MIT does not warrant the Site will

operate in an uninterrupted or error-free manner or that the Site is free of viruses or other harmful components. Use of information obtained from or through this Site is at your own risk. Your access to or download of software, information, materials, or data through the Site or any reference sites is at your own discretion and risk and that you will be solely responsible for any damage to your property (including your computer system) or loss of data that results from the download or use of such material or data.

IN PARTICULAR, MIT WILL HAVE NO LIABILITY FOR ANY CONSEQUENTIAL, INDIRECT, PUNITIVE, SPECIAL, EXEMPLARY OR INCIDENTAL DAMAGES, WHETHER FORESEEABLE OR UNFORESEEABLE, (INCLUDING, BUT NOT LIMITED TO, CLAIMS FOR DEFAMATION, ERRORS, LOSS OF DATA, OR INTERRUPTION IN AVAILABILITY OF DATA).

Indemnification

You agree to defend, hold harmless and indemnify MIT, and its subsidiaries, affiliates, officers, agents, and employees from and against any third-party claims, actions or demands arising out of, resulting from or in any way related to your use of the Site, including any liability or expense arising from any and all claims, losses, damages (actual and consequential), suits, judgments, litigation costs and attorneys' fees, of every kind and nature. In such a case, MIT will provide you with written notice of such claim, suit or action.

Miscellaneous

Termination Rights. You agree that MIT, in its sole discretion, may terminate your use of the Site or your participation in it thereof, for any reason or no reason.

Entire Agreement. This Agreement constitutes the entire agreement between you and MIT with respect to your use of the Site, superseding any prior agreements between you and MIT regarding your use of the Site.

Waiver and Severability of TOS. The failure of MIT to exercise or enforce any right or provision of the TOS of Site shall not constitute a waiver of such right or provision. If any provision of the TOS is found by a court of competent jurisdiction to be invalid, the parties nevertheless agree that the court should endeavor to give effect to the parties' intentions as reflected in the provision, and the other provisions of the TOS remain in full force and effect.

Choice of Law/Forum Selection. You agree that any dispute arising out of or relating to these Terms or any content posted to a Site will be governed by the laws of the Commonwealth of Massachusetts, excluding its conflicts of law provisions. You further consent to the personal jurisdiction of and exclusive venue in the federal and state courts located in and serving Boston, Massachusetts as the legal forum for any such dispute.

Effective Date: April 20, 2015.

Properties

AboutScreen

Information about that screen/page. It appears when the user selects "**About this Application**" from the system menu. Use it to tell users about your App. If you have more than one screen, each screen will have its own AboutScreen.

i.e. This App will works out the cheaper of 2 things given the price and amount of each.

AlignHorizontal

Do you want you main screen horizontally aligned Left, Center or Right.

AlignVertical

Do you want you main screen vertically aligned Top, Center or Bottom. This will have no effect if the screen is **Scrollable**.

AppName

This is the display name of your App when installed on a Device. If the AppName is is not filled in the default will be set to the project name when the project is built.

BackgroundColor

Pick a colour, any colour from the following.

If you don't like any of them use **BackgroundImage**.

BackgroundImage

The screen background image.
Pick a .jpg or .png, if it's integral to the App you may have to fix the orientation of the image or as it turns from portrait to landscape button or text might not look correct.

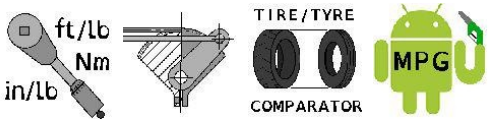
CloseScreenAnimation

The animation for closing current screen and returning to the previous screen. Valid options are default, fade, zoom, slidehorizontal, slidevertical, and none Height Screen height (y-size).

Icon

(designer only) Each of your Apps should have their own Icon, bookmarks on your home screen then you will need to think up some unique Icons 512 pixels by 512 in .png and .jpg format and one 114 pixels by 144 as a .png. I use the .jpg in the App and the other 2 are if you want to put it

on Amazon, free or for sale.



These are some of the ones I've created.

OpenScreenAnimation

The animation for switching to another screen. Valid options are default, fade, zoom, slidehorizontal, slidevertical, and none

ScreenOrientation

The requested screen orientation, specified as a text value. Commonly used values are landscape, portrait, sensor, user and unspecified. See the Android developer documentation for ActivityInfo.Screen_Orientation for the complete list of possible settings. If you use an image as your back ground then you may want this fixed in landscape or portrait so it always looks as you intended it to.

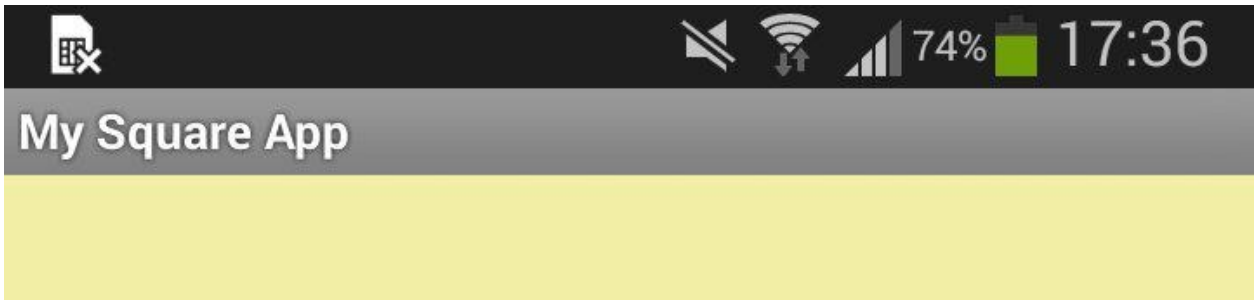
Scrollable

When checked, there will be a vertical scrollbar on the screen, and the height of the application can exceed the physical height of the device. When unchecked, the application height is constrained to the height of the device.

ShowStatusBar

The status bar is the topmost bar on the screen. This property reports whether the status bar is visible. 

Unticked - no Status bar will be shown on your Device.



Ticked - the status bar will be shown.

It will always appear on the Preview screen?

Sizing

If set to fixed, screen layouts will be created for a single fixed-size screen and autoscaled. If set to responsive, screen layouts will use the actual resolution of the device. See the documentation on responsive design in App Inventor for more information. This property appears on Screen1 only and controls the sizing for all screens in the app.

Title

The caption for the form, which appears in the title bar

TitleVisible

The title bar is the top gray bar on the screen. This property reports whether the title bar is visible.

VersionCode

An integer value which must be incremented each time a new Android Application Package File (APK) is created for the Google Play Store.
I start with **10** and then **11** if it's a small change or **20** if it's major.

VersionName

A string which can be changed to allow Google Play Store users to distinguish between different versions of the App.
In the previous cases it would be **1.0** and then **1.1** or **2.0**.

If my VersionCode is **15** then my VersionName will be **1.5**, you don't have to follow this way of working but it works for me.

And Finally

I'd like to thank Carole my long suffering partner for putting up with my hours spent in front of the PC or Device hoping I'll get on with something worthwhile, whilst I'm either working on another App, Book or Website!

Well that about wraps it up for this first Edition of Creating Calculating Android Apps using **MIT App Inventor 2** and hopefully it will help you creating yours. If you find any errors or have any questions or thoughts you'd like to share. Contact me on createapps@omniauto.plus.com and I will try and answer all emails and add any new info in the next edition if needs be. Thanks for reading this. Don't forget to let us know what you thought of the book and if you liked it please click the "liked" on Amazon.

If you do put your Apps on Amazon or Google, don't be put off by Reviews mine range from "useless" which in itself was useless as it says nothing about what they thought was useless to "What I wanted. Simple to use, no extra crap. Key in your number and get the conversion in all formats. That's it that's all. Refreshing for a change. Thanks to the developer for that. Too much software these days is overdone to the point of being unusable."
You'll have to develop a thick skin and if valid points are made they can be used to make the App better.
Don't forget to check out other Apps before you spend hours or days making one when a good one already exists and might be free. Download similar Apps to ones you want to create to see what others have done, you may see pitfalls in theirs that will help yours to be better.

If you found any of this useful you may enjoy my other books on Html Code, Kindles and Java to name a few just click on **Nik Handford** Author on the book page or put **nik handford** into the Amazon search window you may find something else useful amongst them.

Also check out our Apps on Amazon by searching **apps omniautos**, lots are free and may give you more of an idea for your own Apps.

Other Books Available;

PC Basics
Alt, Control, Mouse, PrtScn, Space, Tab, button commands and hints on how to use your PC/Laptop more easily.

WebWriter's Pocket Reference Guide
Is set out as a dictionary with commands and phrases with examples of how to use them, and switches or sub-commands within that command.

Easy Web Writing - 2nd Edition
Including How to Write Web Pages, Sites and Even Kindle Books

Android Tablet Basics - 3rd Edition
How to do the most basic things on Android and some more difficult ones. Often Number 1 in the Computer Literacy list on Amazons.

Java - Questions and Answers
- inputs, calcs and outputs
Or how to ask the user for an Input, take that Input into an equation and print out an Answer.

JavaScript - Questions and Answers
- inputs, calcs and outputs
Or how to ask the user for an Input, take that Input into an equation and print out an Answer in an Html page.

PC Health
Hints and Tips for a Faster Machine or how to keep your PC or Laptop working like it did when you bought it, if not better.

Kindle Answers
2nd Edition
How to:
get the most out of it, write Kindle books, Phone number/Address List, Photo Albums. Emailing it. Find things on it and loads more.

Tyre/Tire Markings
What do they Mean?
Now also available in Paperback.

Books on Classic Cars.