


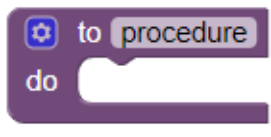



Unit 6 Do-result 介紹

作者：蕭志翔

本章介紹

本章主要介紹 Do-result 指令與 Procedure do、Procedure result 兩種副程式的介紹及使用方法，並讓使用者可以透過本章的教學範例，輕鬆學習，在指令的運用上更加的得心應手。

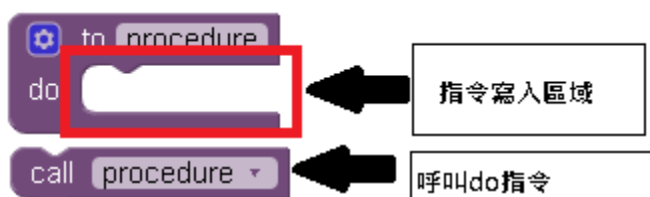
認識本範例使用的拼圖和其基本定義

Control 指令區	
	用來執行任務 (do) 及回報任務結果 (result)，將要運行的拼圖放入 do 執行，並回報一個 result 回程式中。
Procedures 指令區	
	將多個程式指令寫在 Do 中，之後透過另一個呼叫指令(call)來使用這個程序。
	用來呼叫 Procedure do 指令。
	用來設置一個結果(result)，之後透過呼叫來使用它。
	用來呼叫 Procedure result 指令。

而在介紹 Do-result 指令前，讓我們先學習 Procedure do、Procedure result 兩種副程式的定義及使用方法，並了解他們三者間的不同吧。

Procedure do 介紹

Procedure do 副程式是一種程序，你可以將多個指令寫進 Do 中，並在需要使用时，在透過 call 指令來呼叫該程序。

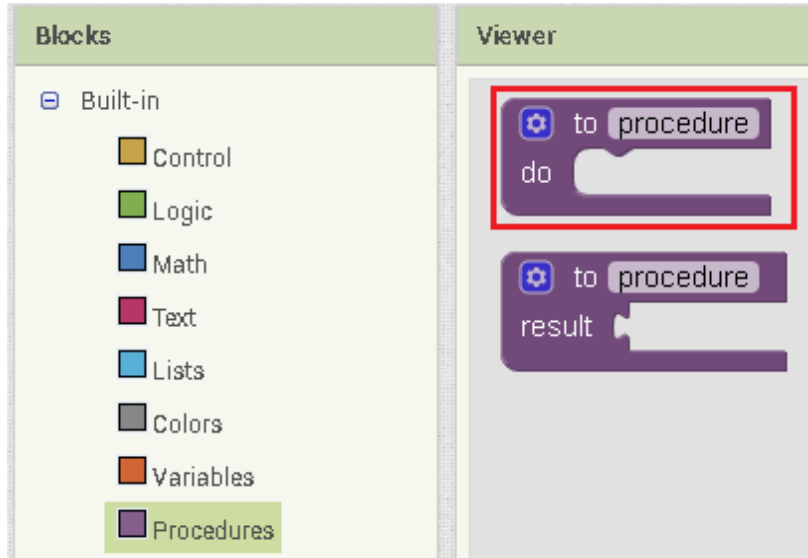


而 Procedure do 與 Do result 不同的地方是，我們需要用 **call 指令** 來呼叫

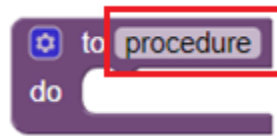
Procedure do，它才會開始執行 do 中的指令集，就像是等基地先發送指令 (call)，部隊才會開始行動 (Do)。

Procedure do 程序方塊設定

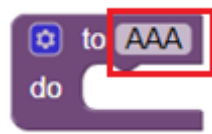
Procedure do 方塊是在內建方塊(Blocks)項目點程序(Procedures)，選取 **to procedure do** 拼圖。



接下來可更改方塊中的 procedure 名稱，例如改為 AAA，表示使用一個名為 AAA 的程序。

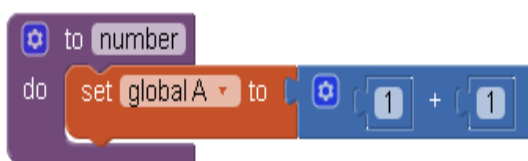


(更改名稱前)



(更改名稱後)

你可以透過更改名稱，來為你的程序設定它的名子，而當你有多個程序時，你能透過你所設定的名稱，來方便尋找需要的程序，例如：有關數字的程序取名為 **number**，計算答案的程序取名為 **Answer**，彈出 hello 視窗的程序取名為 **sayHello ...**。



(number 程序，設定全域變數 A 為 1+1)



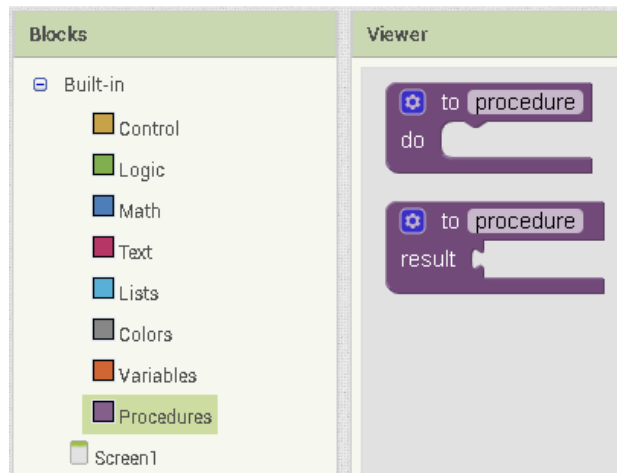
(Answer 程序，設定 Label 的文字顯示為 A 全域變數)



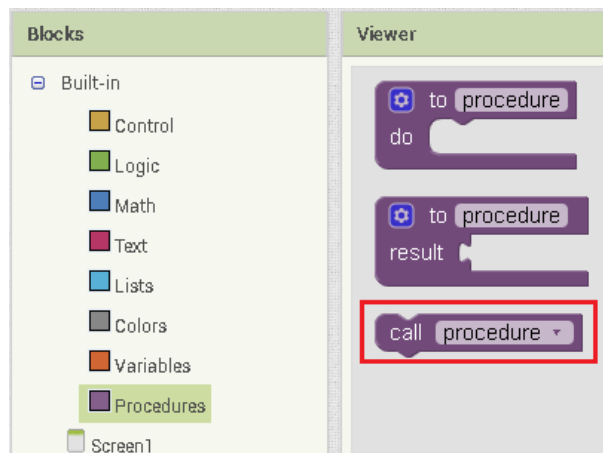
(sayHello 程序，設定 Label 的文字顯示為 Hello!)

呼叫 Procedure do 程序方塊

當我們程序設定完成後，我們可以透過呼叫指令(call)拼圖，來使用這些程序。要注意的是，呼叫指令(call)拼圖，是當我們新增一個程序時，它才會出現的。

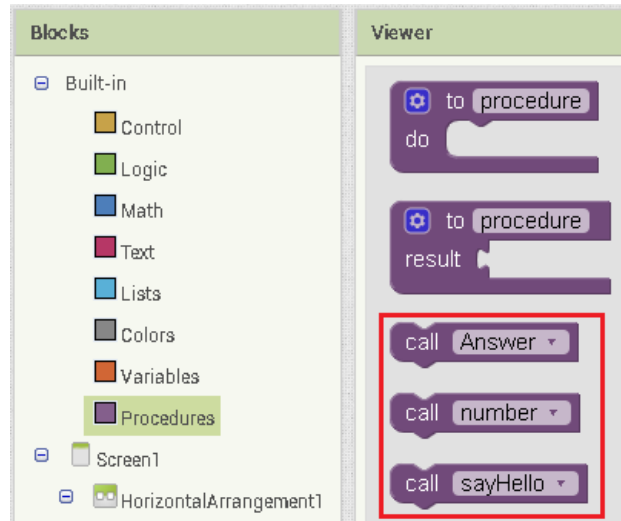


(新增一個程序前)

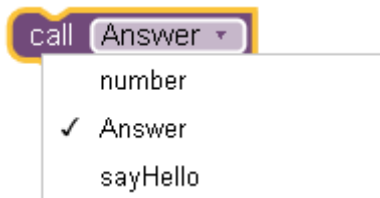


(新增一個程序後)

你新增幾個程序，程序(Procedures)中就會多幾個呼叫指令(call)拼圖。



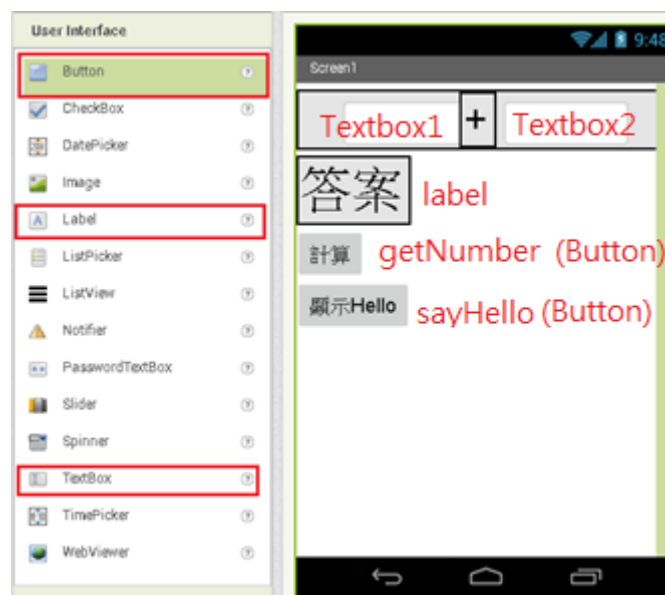
而你也可以使用下拉式選單，更改你所要呼叫的程序。



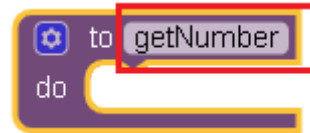
範例說明

請跟著範例做出一個由使用者輸入的簡單加法計算，並設置一個 Label 用來顯示、兩個 TextBox 讓使用者輸入數值、兩個按鈕(Button)一個可以計算答案、另一個則可以讓螢幕顯示 " Hello ! " 。

首先我們先將版面設置好，將上面所需要的物件拉至版面排版，而版面的配置依個人喜好編排即可。

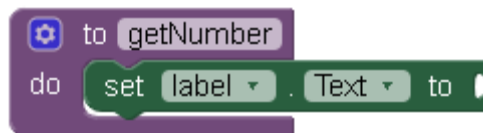
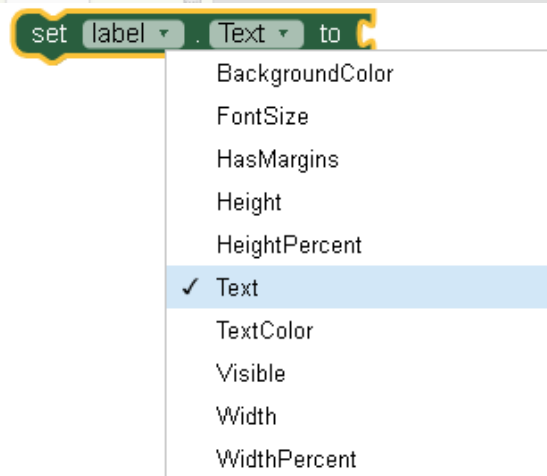
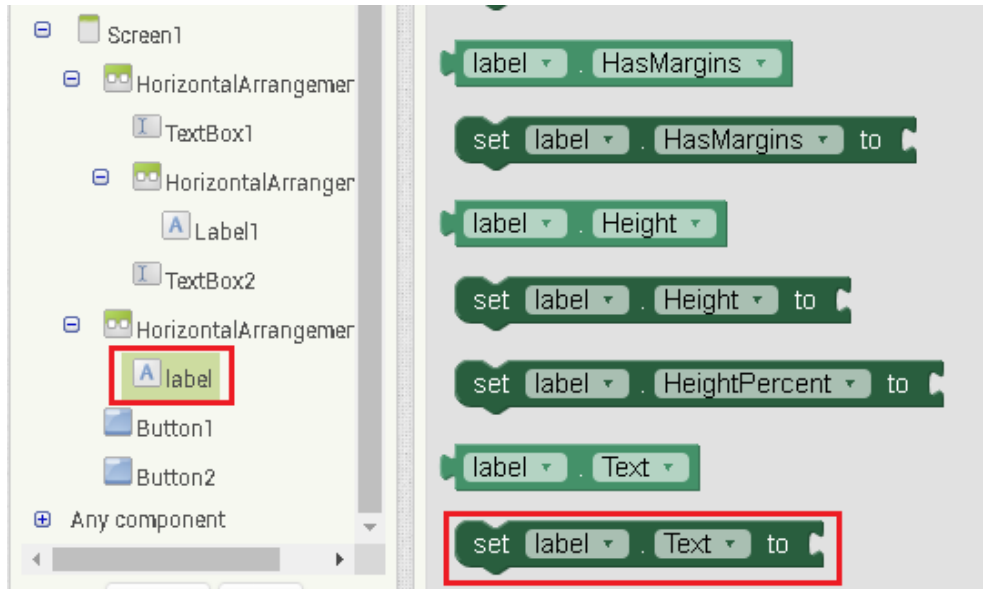


程式方面，先新增一個程序，並將它取名為 `getNumber` 。

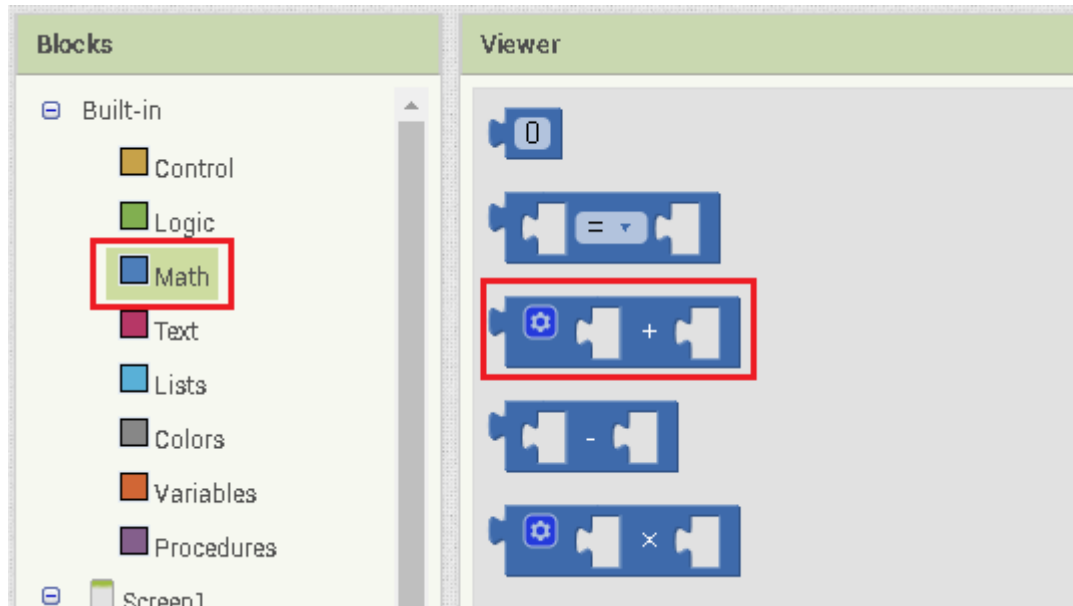


並在內建方塊(Blocks)項目中，找到剛剛新增的 Label 並將拼圖 **set Label.Text to** 組合到程序拼圖當中。

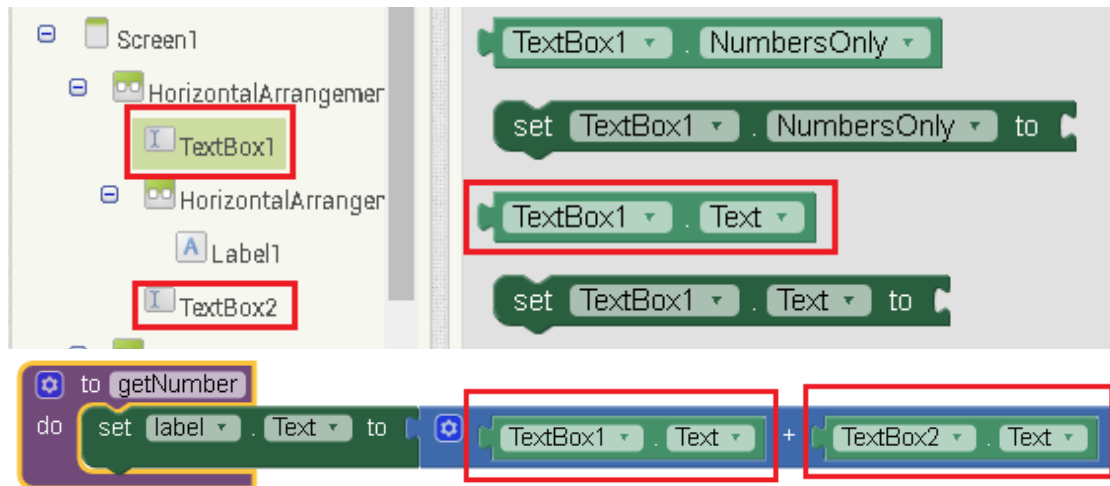
set Label.Text to 是來讓 Label 顯示文字，你也能用下拉式選單更改其他屬性。



接下來一樣在內建方塊(Blocks)項目中找到 Math，並將加法拼圖組合到程序拼圖當中。



最後將兩個 TextBox 的 **set TextBox.Text to** 組合進程序中的加法拼圖裡，這樣計算答案的程序拼圖就完成囉。

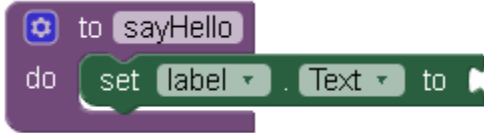


接下來我們設定一個程序名為 sayHello，用來當我們按下按鈕時，可以呼叫該程序來讓 Label 顯示 "Hello!"。

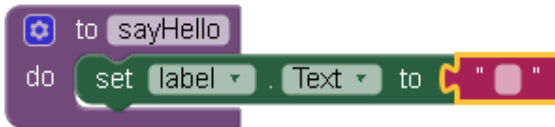
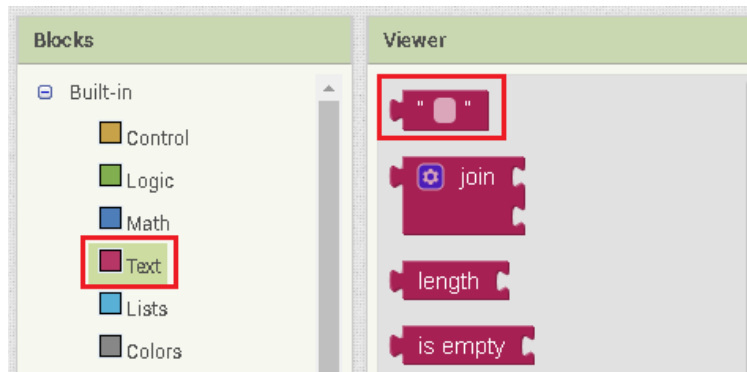
首先新增一個程序，並取名為 sayHello。



與剛剛一樣，新增拼圖 **set Label.Text to** 組合到 sayHello 程序拼圖當中。



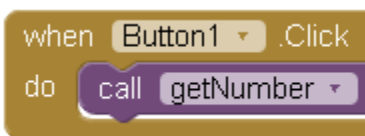
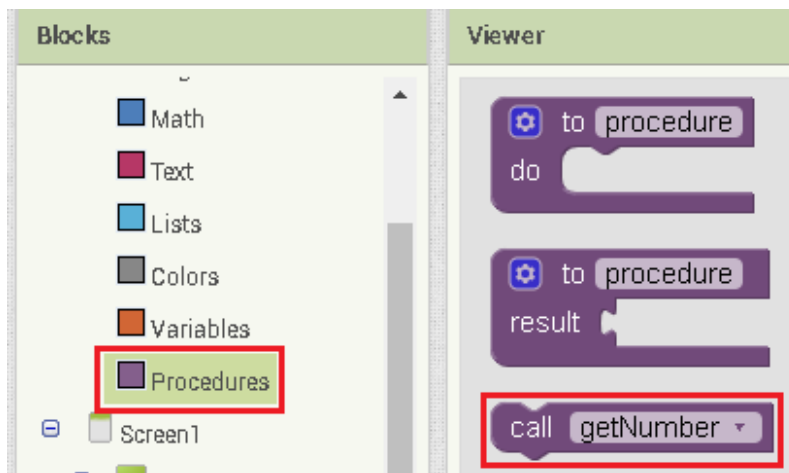
並新增 Text 中最上面的空白字串到 sayHello 程序拼圖，並輸入”Hello！”。



最後從內建方塊(Blocks)項目中找到 Button，並將裡面的 **when button.Click.do** 拼圖加到程式中。



接下來從內建方塊(Blocks)項目的程序(Procedures)中，選取 **call getNumber** 拼圖，將它與 **when button.Click.do** 組合。



這樣當使用者按下 Button1 的按鈕時，他就會呼叫 getNumber 這個程序了並照著一樣的方式設定，當 Button2 按下時呼叫 sayHello 這個程序。



這樣程式就範例就完成囉，來觀看測試結果吧。



(初始畫面)



(輸入數字計算後)



(點下 Hello 按鈕)

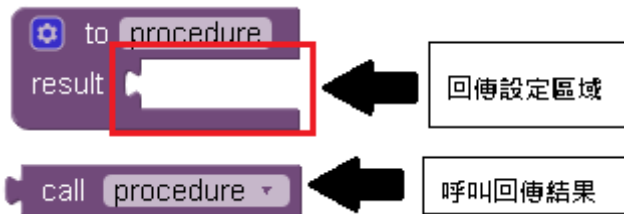
活 動

請修改上面的例子，將當中的加法變成減法，並設定當按下 Hello 按鈕後，顯示自己的名子。

Procedure result 介紹

Procedure result 副程式也是一種程序，它會回傳一個使用者設定的結果 (result)，在透過呼叫(call)指令來呼叫該程序。

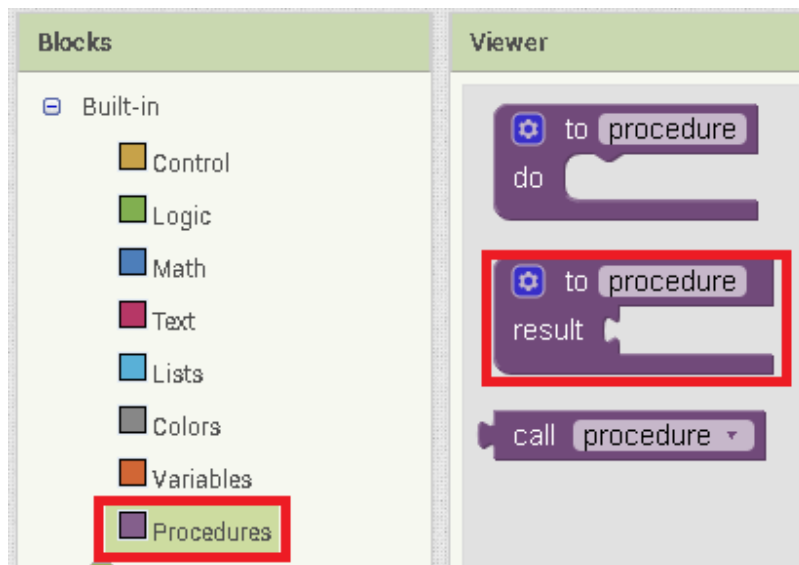
簡單來說，就是當基地呼叫部隊時 (call)，部隊則是直接回報任務結果 (result)，而不是執行任務 (do)。



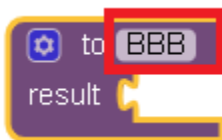
與 Procedure do 不同的地方是，當使用 Procedure result 時，它會直接回傳一個結果(result)，而並非將程式拼圖寫入 do 中，來執行程序結果。

Procedure result 方塊設定

Procedure result 方塊在內建方塊(Blocks)項目點程序(Procedures)，選取 **to procedure result** 拼圖。



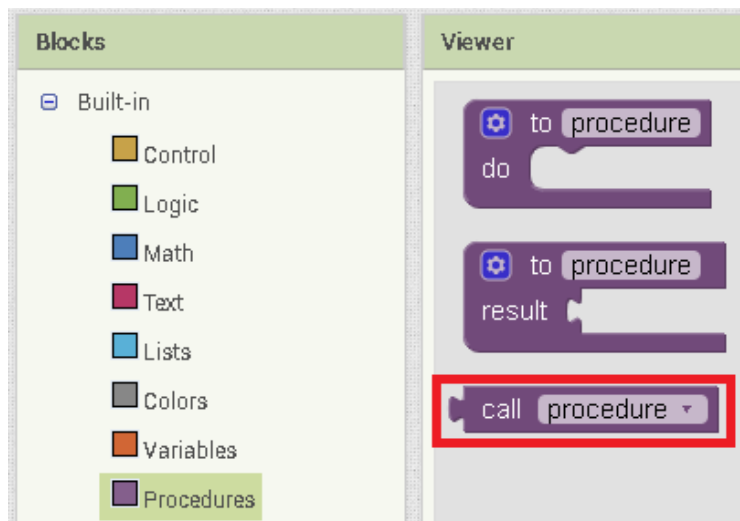
與 Procedure do 方塊一樣，你可以更改名稱來分辨不同的程序。



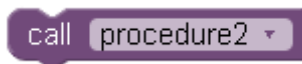
呼叫 Procedure result 程序方塊

與 Procedure do 方塊一樣，當我們設定完 Procedure result 方塊後，我們可以透過呼叫指令(call)拼圖，來使用這些程序。

而呼叫指令(call)拼圖，一樣是當我們新增一個完 Procedure result 後，它才會出現的。



這裡需要注意，當使用者的程式同時擁有 **call Procedure do**、**Procedure result** 兩種方塊時，你可以透過拼圖形狀的不同，來分辨它們。



(call Procedure do 拼圖)

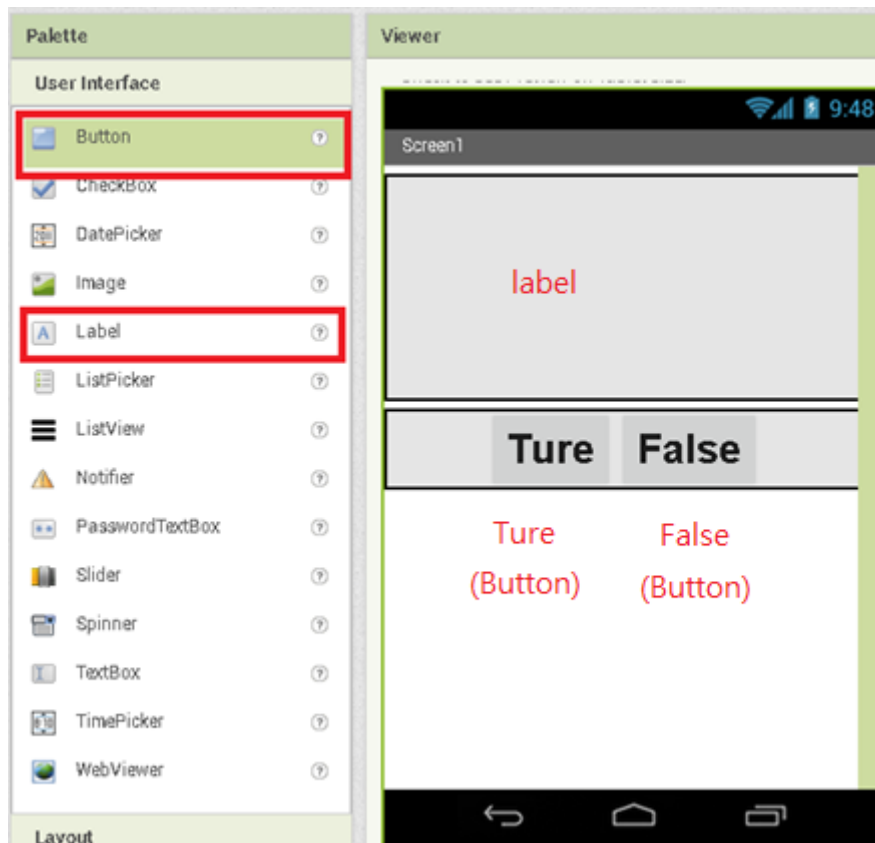


(call Procedure result 拼圖)

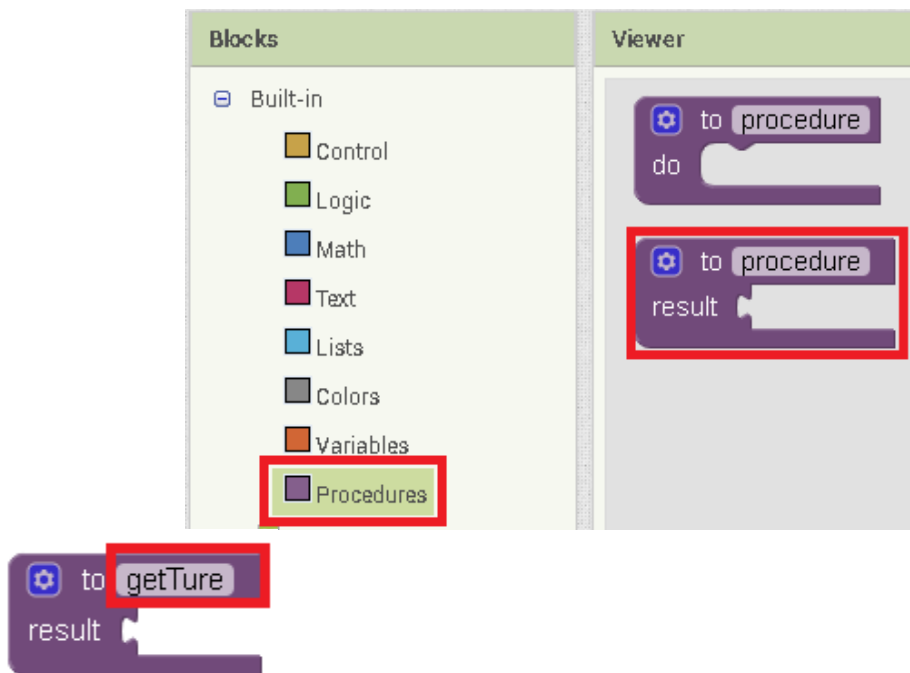
範例說明

請跟著範例做出一個隨著使用者按下兩個不同的按鈕(Button)時，會回傳 true 及 false 兩種結果，並顯示到 Label 的簡單程式。

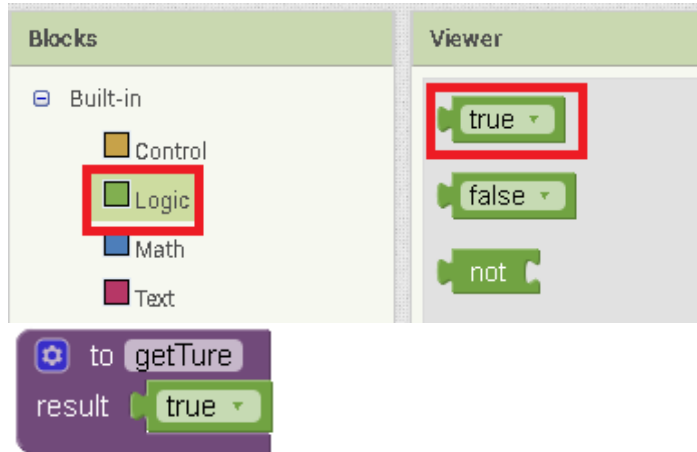
首先將上面所需的物件拉至版面排版，而版面配置一樣依個人喜好編排即可。



程式方面，先至內建方塊(Blocks)項目點程序(Procedures)，選取 **to procedure result** 拼圖，並將它取名為 **getTure**。

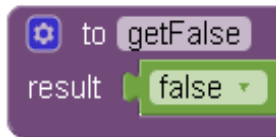


之後至內建方塊(Blocks)項目點邏輯(Logic)，選取 **ture** 拼圖，並將它們組合。



這樣代表當 getTure 會回傳 true 值給程式。

接著照著相同的方式設置一個 getFalse 的 **to procedure result** 拼圖，並將它與邏輯(Logic)中的 **false** 拼圖組合起來。



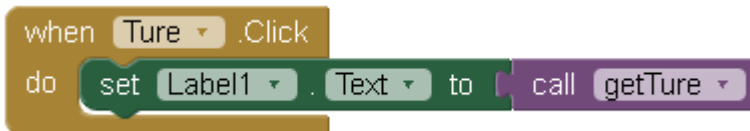
最後從內建方塊(Blocks)項目中的 Button，將裡面的 **when button.Click.do** 拼圖加到程式中。



再將內建方塊(Blocks)項目中的 Label，**set Label.Text to** 組合到 **when button.Click.do** 拼圖當中。



並再 **set Label.Text to** 後組合上一個呼叫 **getTrue** 程序(call)的指令

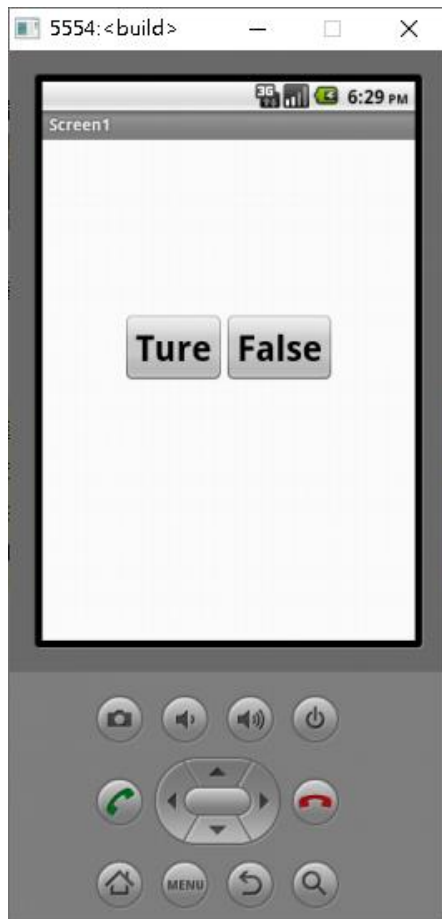


這樣 Ture 按鈕的指令就設置好了。

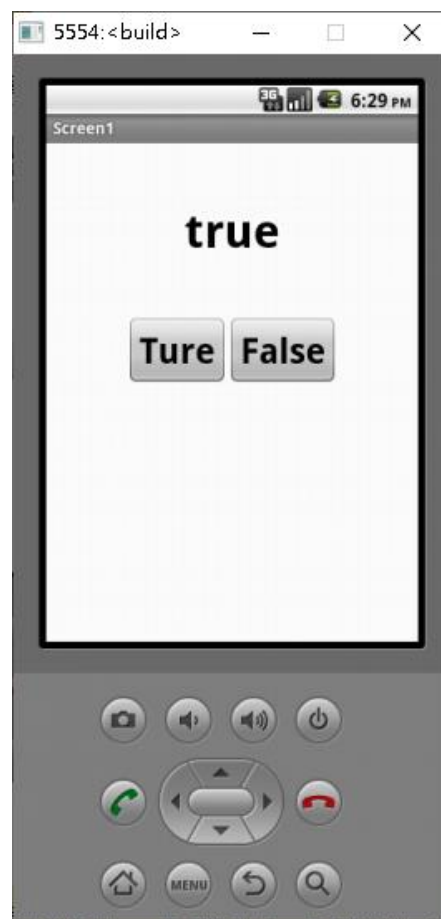
接下來用一樣的方式設置 False 按鈕指令。

```
when False .Click  
do set Label1 . Text to call getFalse
```

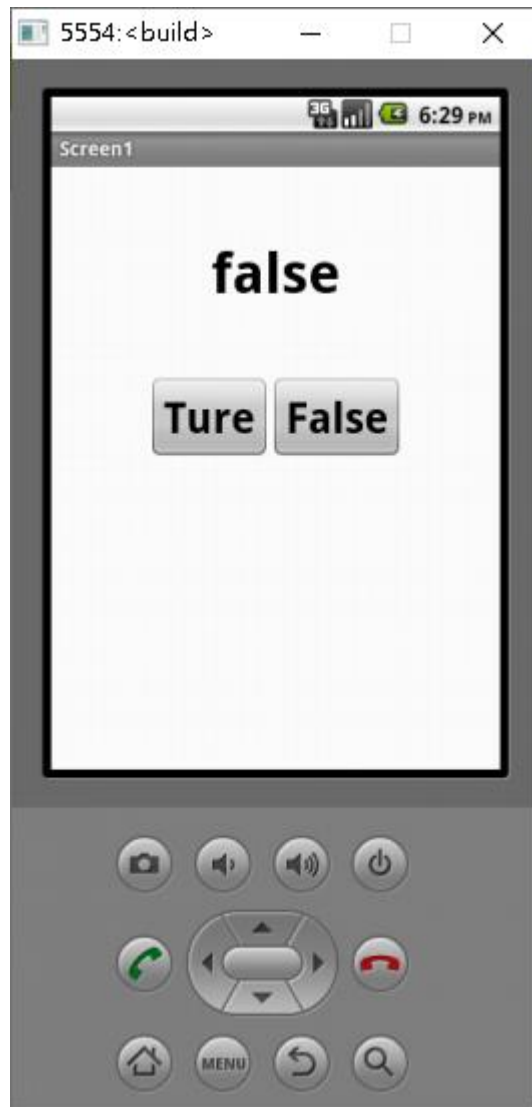
這樣就完成囉，最後觀察測試結果。



(初始畫面)



(按下 Ture 按鈕時)



(按下 False 按鈕時)

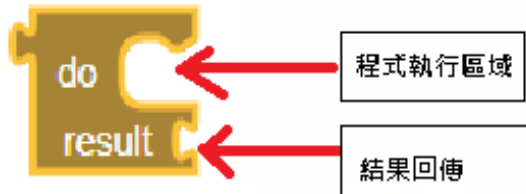
活 動

請修改上面的例子，將範例程式修改成一個電燈開關，並設定按下 ture 為開燈，false 為關燈，再將背景顏色隨開關燈更改（開燈為白色，關燈為黑色）。

* 小提示：可以利用 if 條件式加變數，隨使用者按下的按鈕不同，更改背景顏色

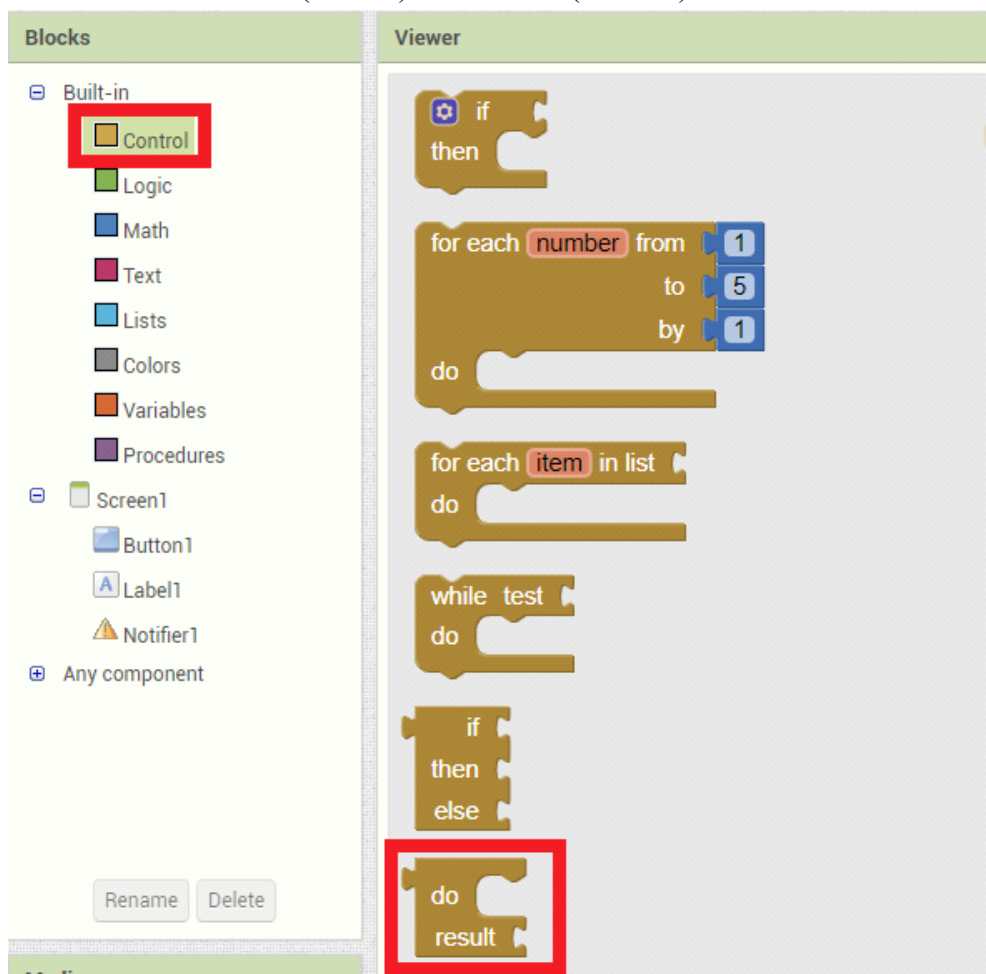
Do result 介紹

Do result 就像將執行任務 (do) 及回報任務結果 (result) 兩件事情結合起來一起做，也就是上面所提到的 procedure do 及 procedure result 兩個指令的組合體，同時達到兩者的效用。



Do result 方塊設定

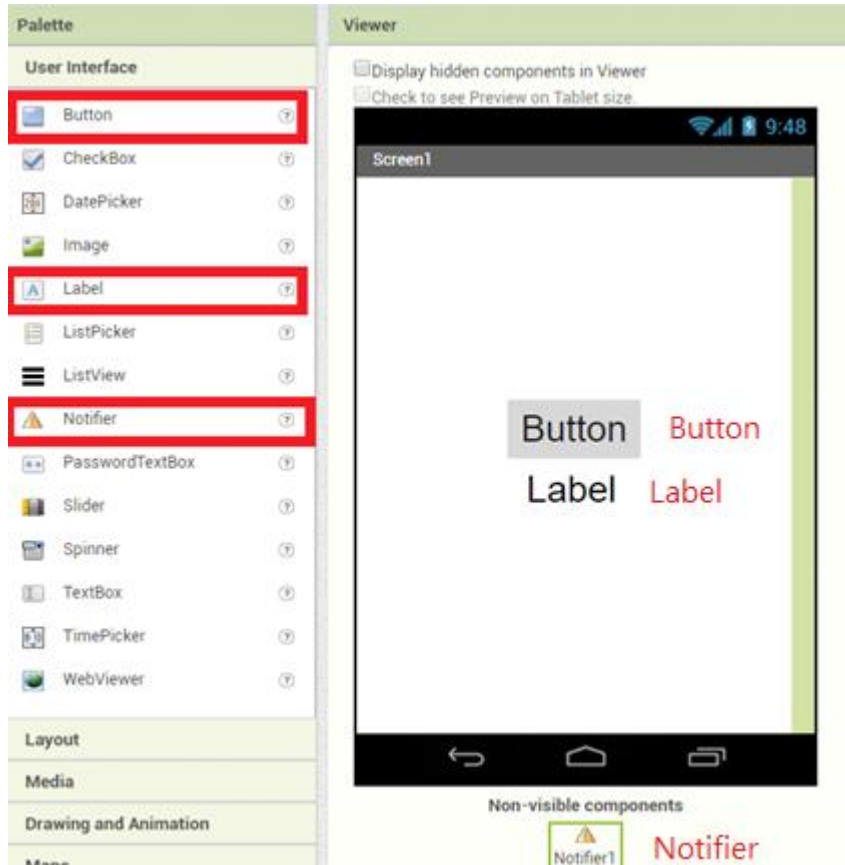
Do result 方塊在內建方塊(Blocks)項目點控制(Control)，選取 **Do result 拼圖**。



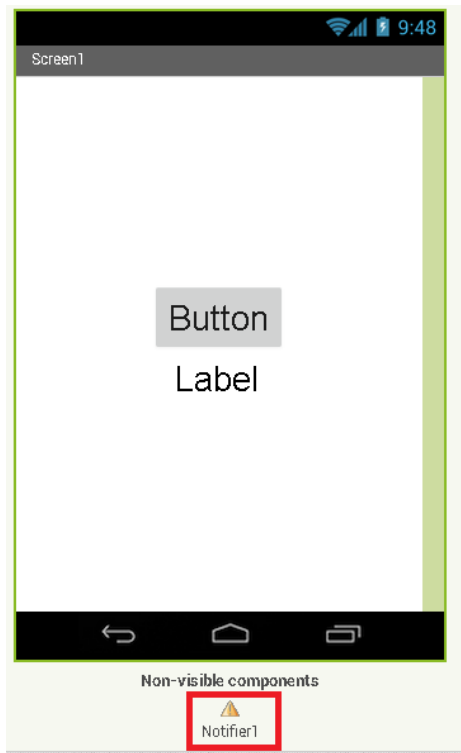
範例說明

我們再畫面新增 Notifier 及 Button 和 Label 三個物件，使我們點擊 Button 按鈕後，可以呼叫 Notifier 視窗，並回傳 true 顯示在 Label 上。

首先將上面所需的物件拉至版面排版，而版面配置一樣依個人喜好編排即可。



這裡可能大家沒看過 Notifier 物件，Notifier 物件是一個彈出式視窗，當程式觸發後，畫面上會跳出一個視窗給使用者看到，而 Notifier 物件在版面上為最下方的小三角形，平常在 App 畫面中是看不到的。



程式方面，先從內建方塊(Blocks)項目點控制(Control)，新增一個 **do result** 指令

方塊及 Notifier 的 `Notifier.ShowDialog`，並將兩者組合起來。

The image consists of three screenshots from the App Inventor 2 IDE, illustrating the assembly of a `Notifier.ShowDialog` block within a `do result` block.

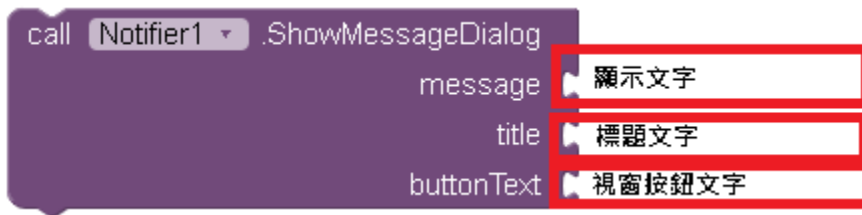
Top Screenshot: Shows the 'Blocks' palette on the left with the 'Control' category highlighted in red. The 'Viewer' on the right shows a 'do result' block highlighted in red, indicating it is the target for the next block.

Middle Screenshot: Shows the 'Notifier1' component highlighted in red in the 'Blocks' palette. In the 'Viewer', a 'call Notifier1.ShowDialog' block is highlighted in red, showing its parameters: message, title, buttonText, and cancelable (set to true).

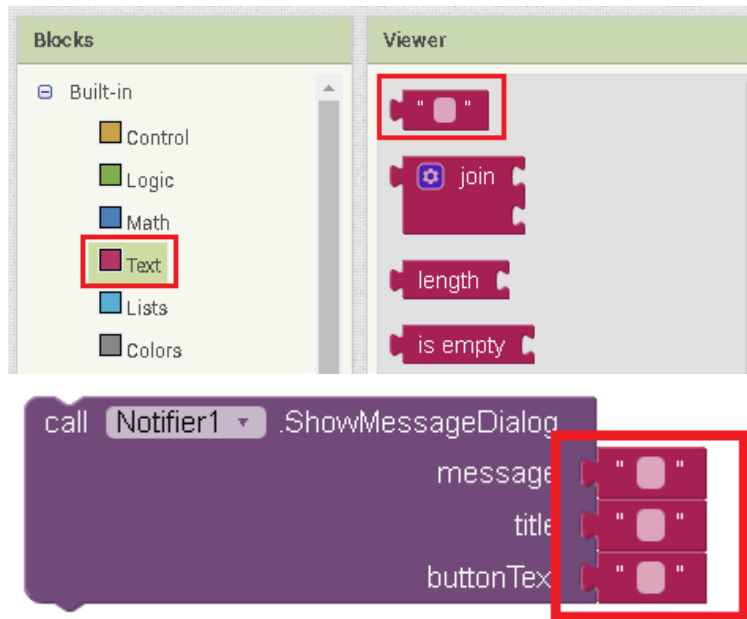
Bottom Screenshot: Shows the final assembly where the 'call Notifier1.ShowDialog' block is placed inside the 'do result' block.

而 `Notifier.ShowDialog` 指令方塊依順序的方塊缺口為視窗顯示文字、

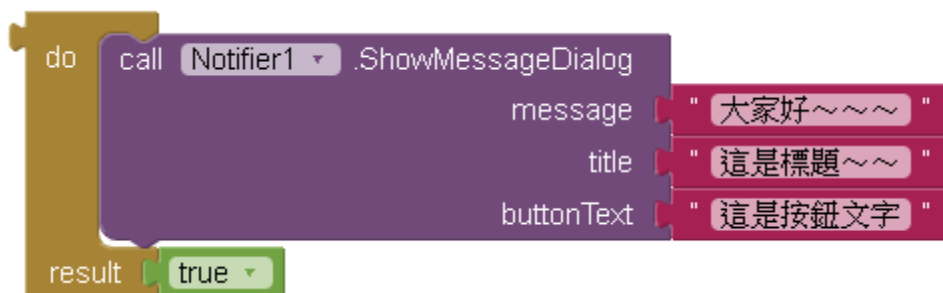
視窗標題文字、視窗按鈕文字。



我們從內建方塊(Blocks)項目點文字(Text)，選取最上方的空白字串，新增至 **Notifier.ShowMessageDialog** 指令方塊的三個缺口。



最後分別輸入 " 大家好 ~ ~ ~ "、" 這是標題 ~ ~ "、" 這是按鈕文字 "，並在 result 後加入 **true** 方塊。



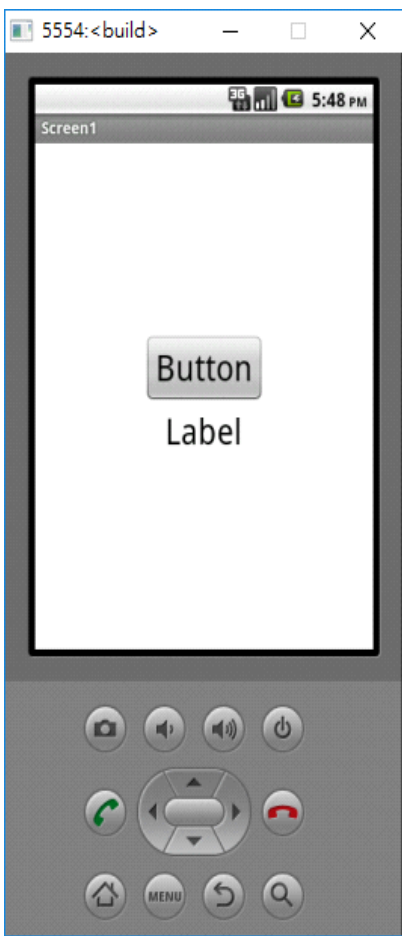
接下來從內建方塊(Blocks)項目中的 Button，將裡面的 **when button.Click.do** 拼圖加到程式中，並與 Label 中的 **set Label.Text to** 組合。



最後將兩個大拼圖組合起來，程式就完成囉。

```
when Button1 .Click
do
  set Label1 .Text to
do
  call Notifier1 .ShowMessageDialog
  message "大家好~~~"
  title "這是標題~~~"
  buttonText "這是按鈕文字"
result true
```

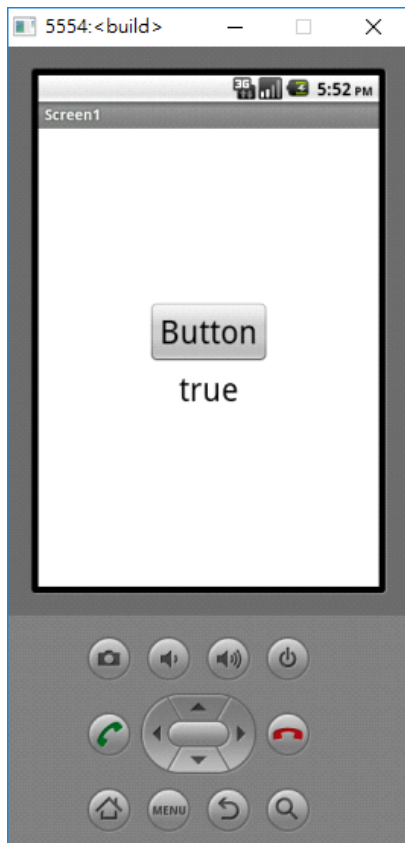
最後觀察測試結果。



(初始畫面)



(點擊按鈕後 · 彈出視窗)



(關閉視窗後，顯示 true)

活 動

請修改上面的例子，將當中的 `Notifier.ShowDialog` 指令方塊依學號、姓名、班級顯示出來，並在 Label 上顯示 `False`。

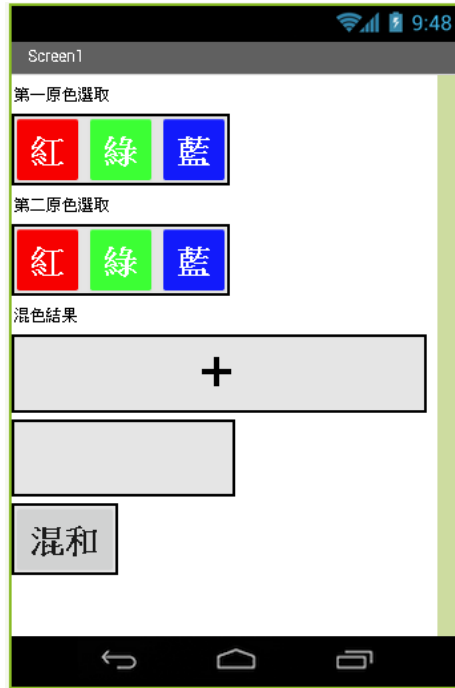
練習題

1. 新增一個 Label(空白)及 Button(開啟世界) 如下圖)，並試著用 Procedure do、Procedure result 2 種指令，各做出一個顯示 hello world 的程式。



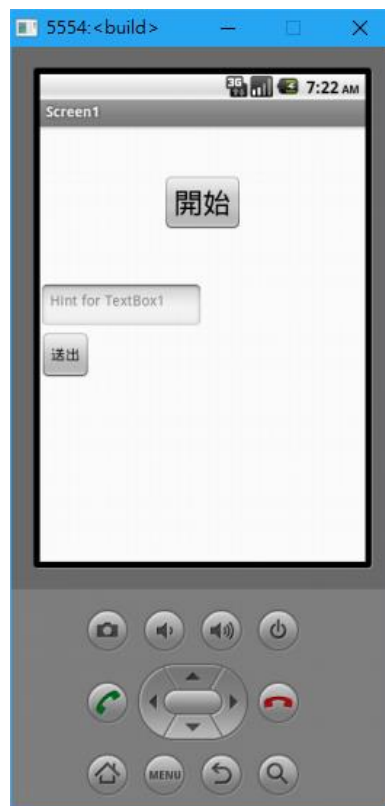
(練習題 1)

2. 呈上題，如果第一題成功，試著用 Do result 寫出一樣的程式。
3. 試著用 Label、Button 設置版面，並用 Procedure do、Procedure result、Do result 指令，做出 RGB 三原色的任兩種原色混和，且將其流程及結果顯示在畫面上。
 (紅 + 綠 = 黃 紅 + 藍 = 粉紅 藍 + 綠 = 青藍
 紅 + 紅 = 紅 藍 + 藍 = 藍 綠 + 綠 = 綠)



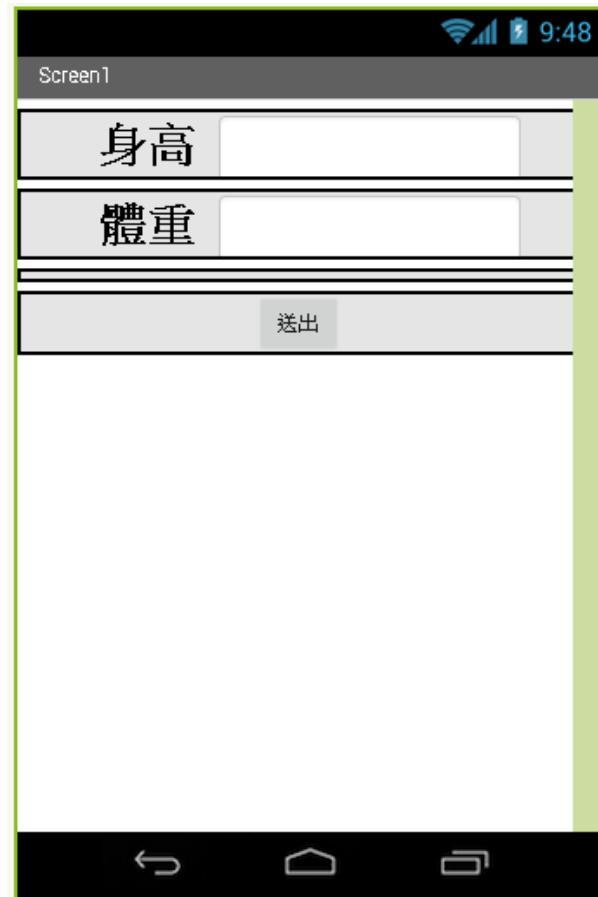
(練習題 3)

4. 試著用 Label、Button、TextBox 設置版面，並用 Procedure do、Procedure result、Do result 指令，做出簡單的加法小遊戲，設定亂數 A、B 為 1~9，且將 A、B 相加，並由使用者回答答案，答對則加 1 分，答錯則扣一分。



(練習題 4)

5. 試著用 Label、Button、TextBox 設置版面，並用 Procedure do、Procedure result、Do result 指令，做出一個計算 BMI 值的程式。(18.5 > BMI = 過輕 24 > BMI >= 18.5 = 正常 24 <= BMI = 過重)



(練習題 5)