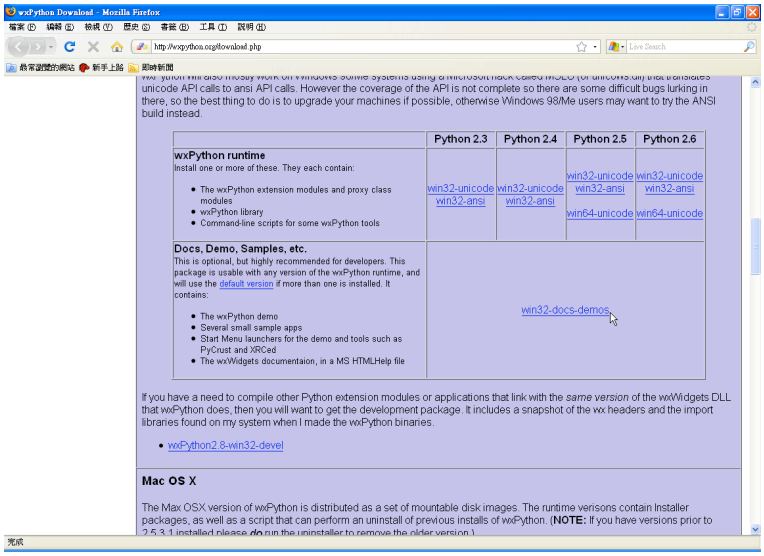
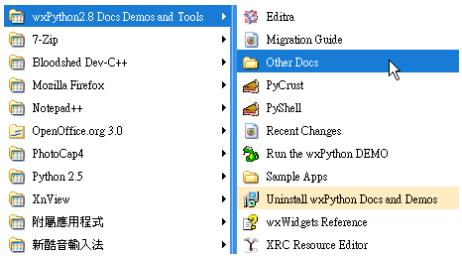
第十五章 Docs、Demo與wxGlade

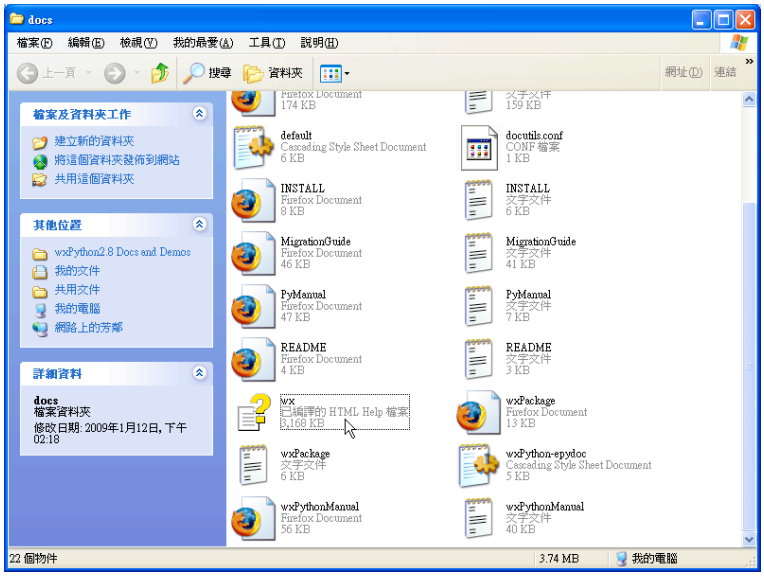
wxPython官網的下載頁，除了wxPython的安裝檔外，還可以下載「Docs, Demo, Samples, etc.」的安裝檔。



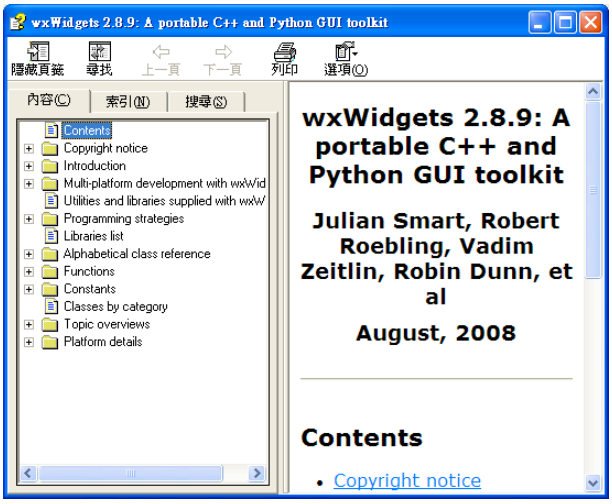
「Docs」就是文件，「Demo」是展示的範例，「Samples」則是範例的原始程式碼，當安裝完成後，打開開始功能表，我們會發現安裝的許多工具程式，先來看看Other Docs資料夾。



裡頭包含安裝指南、手冊等所有的文件，我們移動游標，點擊底下的wx HTML Help說明檔。



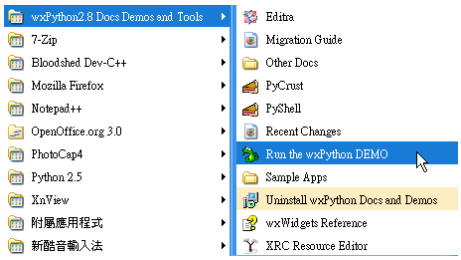
這個說明檔提供了wxWidgets詳盡的參考說明。



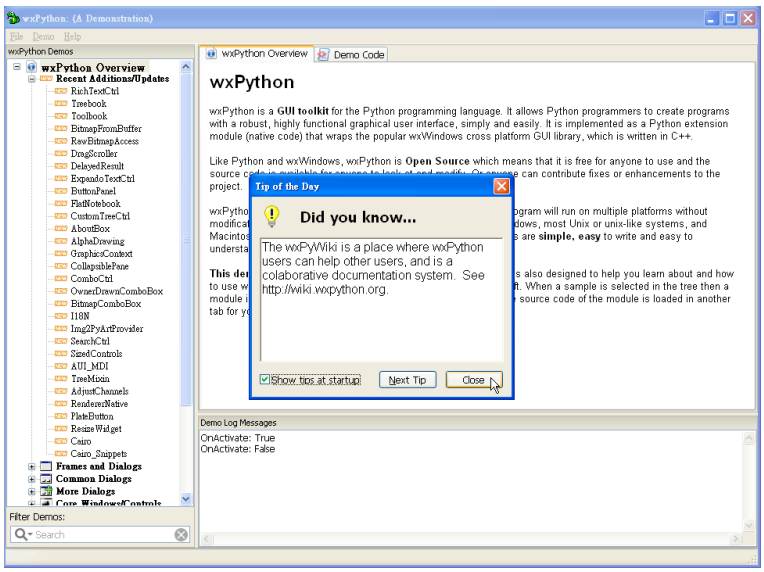
當然，如果要深入研究由wxWidgets延伸而來的wxPython，這些說明文件就會是很重要的參考資源。我們目前不打算深入探討wxPython的強大，倒是可先由Demo來看看wxPython開發軟體的方便與魅力。

Demo

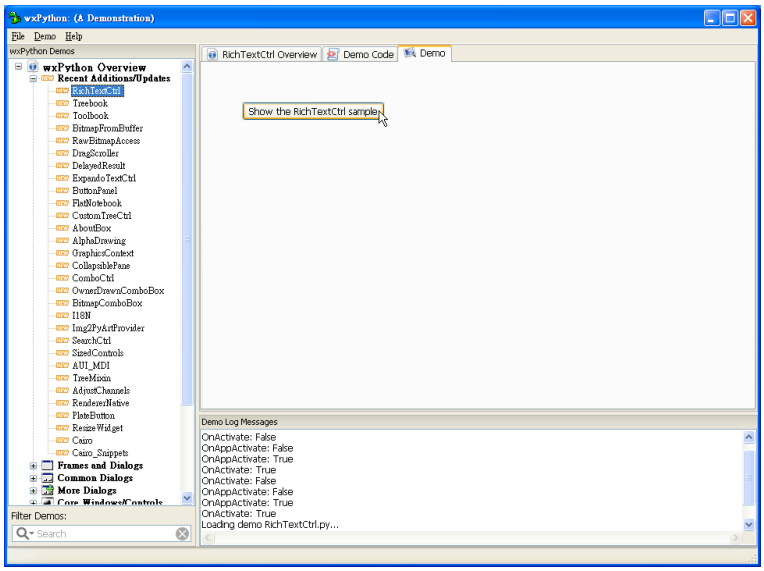
我們回到開始功能表，這次選擇開啟Run the wxPython DEMO。



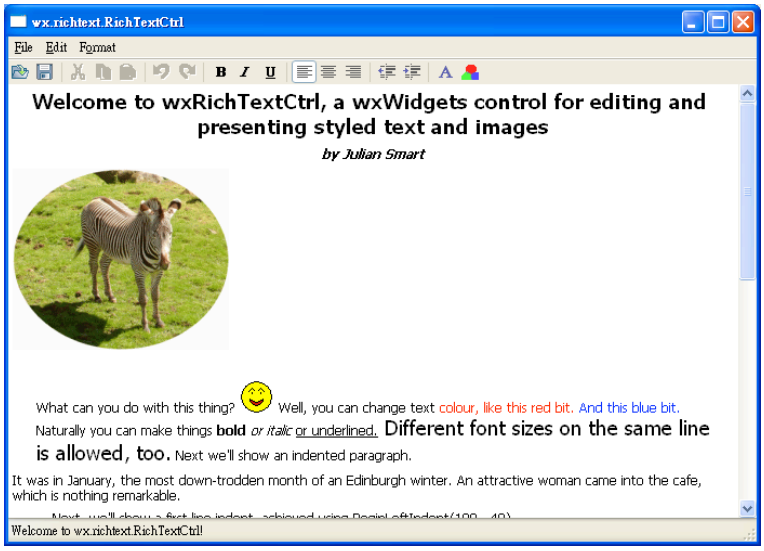
首先印入眼簾的便是「每日小技巧」……



關閉「每日小技巧」的視窗後，視窗顯示的是「wxPython Overview」的內容，我們將其切換到底下的「RichTextCtrl」，然後點選「Demo」分頁，可以看到有個「Show the RichTextCtrl sample」的按鈕，接著點擊這個按鈕，我們來看看這個例子。

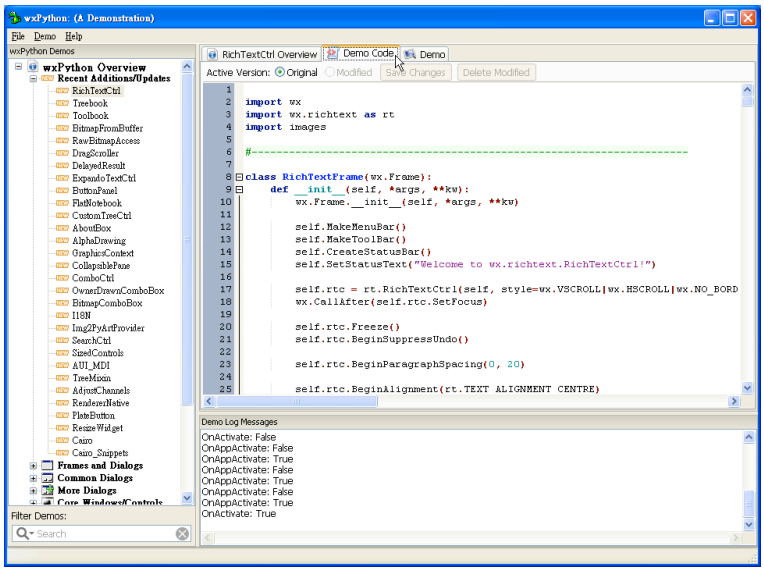


我們可以看到這是個文書處理器的例子，如下圖。



RichTextCtrl裡頭的功能比我們上一章所發展的「簡單的文字編輯器」豐富多了，包含字體、編排、插入圖片等功能一應俱全。這裡擁有許多的例子，有如上具備規模的程式範例，也有非常多屬於視窗元件的個別例子，包含下拉式選單、各式按鈕等等，實際的效果我們都可以自行嘗試執行。

我們若是開啟「Demo Code」分頁，就可以看到該Demo的原始程式碼。



由原始碼看來，RichTextCtrl利用了wx.richtext物件，而圖檔另外利用images模組來處理。我們在這裡不打算詳細說明wx.richtext物件，反倒是要闡明一個觀念，就是「避免重新發明輪子」。

避免重新發明輪子

我們在上一章中以重頭發展的方式，主要目的是逐步分項說明視窗程式設計與wxPython的結合，用為介紹wxPython模組庫的概念，這便是「重新發明輪子」的方式，「輪子」如上一節我們看到的例子，而且似乎比我們發明的輪子運作的更好。既然「輪子」已經有了，為何還要「重新發明」？

以「重新發明輪子」的方式作為學習程式設計的過程，除了好讓我們了解「輪子」是怎麼做出來的之外，更是讓我們容易體會「輪子」如何「滾動」，如何調校可以「滾動」的符合我們的需求。這是一種學習方式，我們同時也都是以這樣方式作為Python教學的主要脈絡。

然而若是我們日後要從事開發工作，或是加入某個開發團隊，譬如團隊的目的是開發文書處理器好了，那麼開發工作也要以「重新發明輪子」的方式進行嗎？不，那樣曠日費時，況且開放原始碼軟體OpenOffice中的Writer或是AbiWord都已經提供了很好的範例，因為我們都能夠下載並閱讀這些軟體的原始程式碼。

這正是開放原始碼的精神與特點，Python本身就是開放原始碼的程式語言，因而有許多延伸的模組庫或是專案，也都是本著開放原始碼的精神，讓我們得以窺探許多精美外表軟體內的程式秘奧，也是我們學習開發程式最佳的範本。

另一方面，程式碼與文字相若，文字可藉由紙張匯集成一本本的書籍，我們取得書籍，不論是自行購買還是在圖書館中借閱，我們都能夠自由閱讀書籍裡頭的內容，若是有多一點的心得，我們還可以延伸進行更多的文字創作。當我們可以自由閱讀軟體程式碼的內容時，除了可以提出精善的意見，同樣的，多一點的心得，使我們可以延伸設計出更好的軟體。

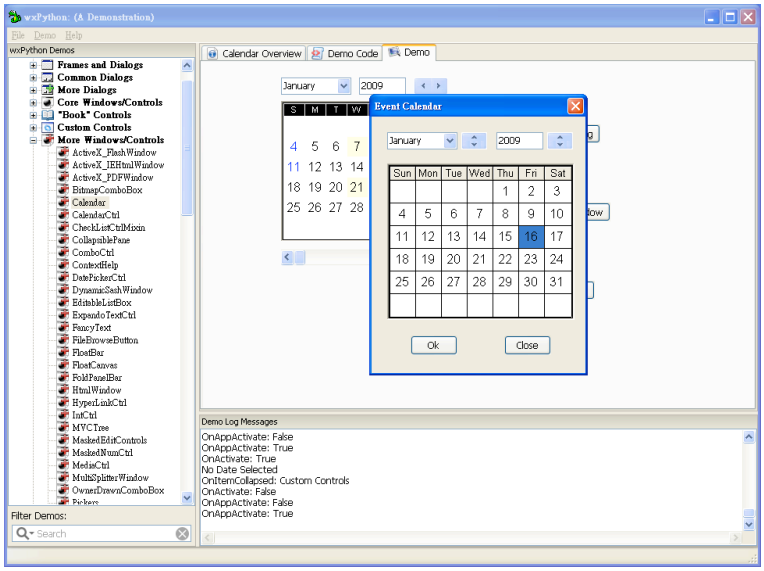
所以，開放程式的原始碼可使我們站在既有的基礎上持續發展，已經由別人做好的東西，我們就不必重頭開發一次，因此，軟體的發展就能夠持續不斷的進行，我們不用因為沒有「輪子」，而被迫反覆的「重新發明輪子」，另一方面，軟體公司也不用陷入「發明輪子」競賽的泥沼之中。

當然，不論使用或是參考開放原始碼的軟體時，我們都需詳讀其授權，不同的授權有其不同的規範，許多常見開放原始碼軟體採用GPL授權，其為最嚴格的授權條款之一，除了要求釋出原始程式碼外，亦不能更改授權方式。中文資訊可參考自由軟體鑄造場的授權條款介紹，以及授權條款比較表，。

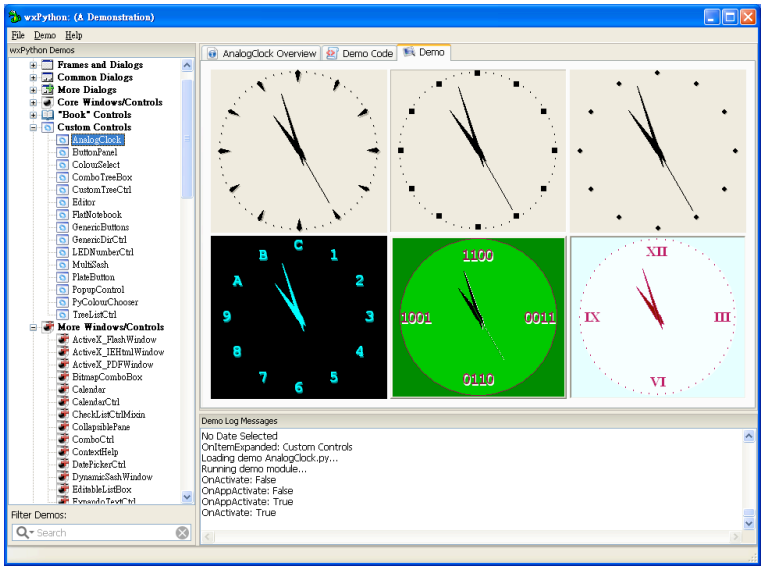
因為要「避免重新發明輪子」，於是我們接下來取Demo中的顯示月曆及時間的程式例子，作為我們新的範例程式的模組庫。Demo採用的是LGPL授權，其主要目的就是作為模組庫（函數庫）的推廣。

先看看效果

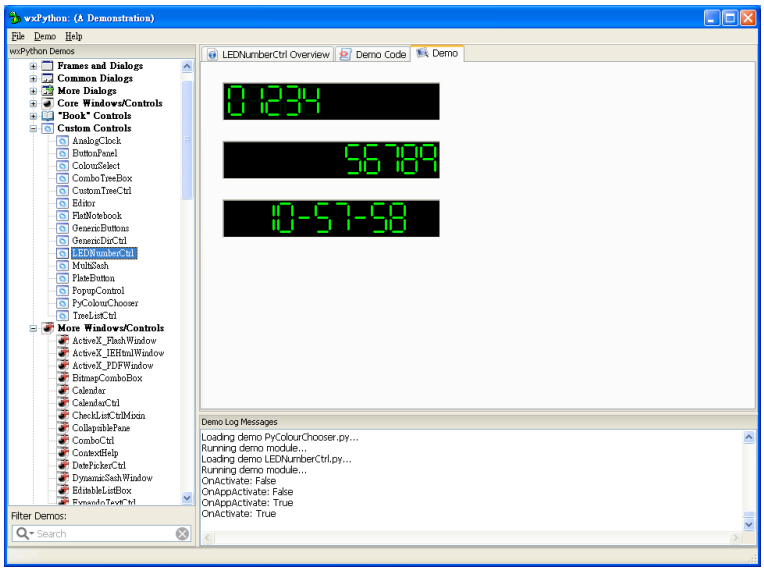
我們從Demo找到了月曆的部份，這可以用另外的視窗顯示出來，如下。



時鐘的部份，Demo有個類比式時鐘的範例，如下。

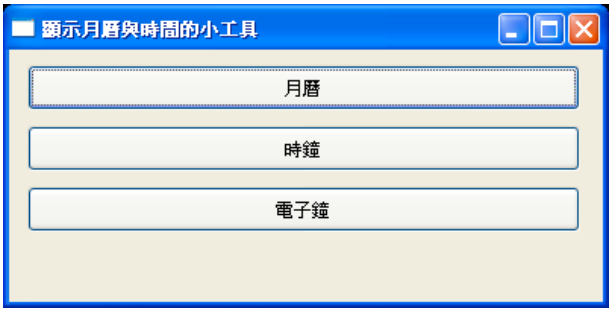


還有個電子式數字與時鐘的顯示，如下。



我們以一個簡單的方式來規劃整合這三個範例，首先把存放範例檔的demo資料夾複製到工作資料夾，如wxtest，然後在wxtest裡的demo資料夾裡加入空白的__init__.py檔案，使demo變成套件可供引入其內的模組。

然後我們以如下的視窗，三個按鈕來個別叫出上述三個範例。



程式碼如下。

```
#-*- coding: UTF-8 -*-

import wx, string

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(400, 200))
        panel = wx.Panel(self, -1)

        font = wx.SystemSettings_GetFont(wx.SYS_SYSTEM_FONT)
        font.SetPointSize(12)

        vbox = wx.BoxSizer(wx.VERTICAL)

        vbox.Add((-1, 10))

        hbox1 = wx.BoxSizer(wx.HORIZONTAL)
        self.button1 = wx.Button(panel, 1, u"月曆", size=(365, 30))
        #self.Bind(wx.EVT_BUTTON, self.ButtonClick1, id = 1)
        hbox1.Add(self.button1, 0)
```

```
            vbox.Add(hbox1, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
            vbox.Add((-1, 10))

            hbox2 = wx.BoxSizer(wx.HORIZONTAL)
            self.button2 = wx.Button(panel, 2, u"時鐘", size=(365, 30))
            #self.Bind(wx.EVT_BUTTON, self.ButtonClick2, id = 2)
            hbox2.Add(self.button2, 0)

            vbox.Add(hbox2, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
            vbox.Add((-1, 10))

            hbox3 = wx.BoxSizer(wx.HORIZONTAL)
            self.button3 = wx.Button(panel, 3, u"電子鐘", size=(365, 30))
            #self.Bind(wx.EVT_BUTTON, self.ButtonClick3, id = 3)
            hbox3.Add(self.button3, 0)

            vbox.Add(hbox3, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
            vbox.Add((-1, 10))

            panel.SetSizer(vbox)
            self.Centre()
            self.Show(True)

        def ButtonClick1(self):
            pass

        def ButtonClick2(self):
            pass

        def ButtonClick3(self):
            pass

    if __name__ == "__main__":
        app = wx.App()
        frame = MyFrame(None, -1, title=u"顯示月曆與時間的小工具")
        app.MainLoop()
```

三個按鈕的程式還沒寫，因為這牽涉到如何運用demo模組庫的功能，我們需要審察一下demo模組庫。

整合問題

這裡，我們把demo資料夾改成了一個模組庫，所以我們能夠由其內引入需要的模組，而顯示月曆、類比式及電子式時鐘分別儲存在Calendar.py 、 AnalogClock.py 、 LEDNumberCtrl.py三個檔案中，實際上就是Calendar、AnalogClock、LEDNumberCtrl三個模組。

所以在新的程式先要引入這三個模組。

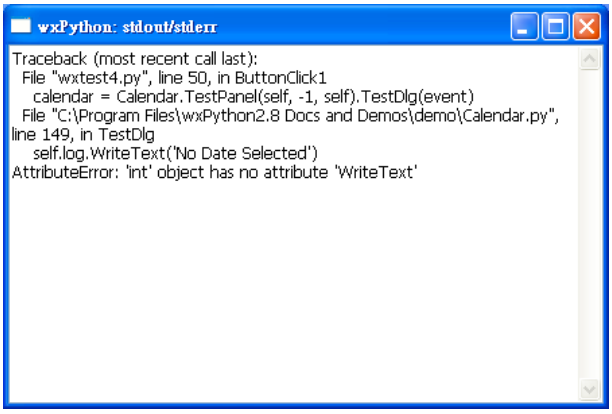
```
from demo import Calendar, AnalogClock, LEDNumberCtrl
```

月曆的部份，由於是利用Calendar模組的TestPanel型態，其內有TestDlg()方法叫出顯示月曆的視窗，因此我們要將原先MyFrame型態中ButtonClick1()方法的pass陳述移除，接著改成如下的內容。

```
def ButtonClick1(self, event):
    calendar = Calendar.TestPanel(self, -1, self).TestDlg(event)
```

記得要將建立「月曆」按鈕後註解化的「#self.Bind(…)」的地方移除註解，然後我們重新執行程式，點擊月曆的按鈕，可得以下的結果。

但是當我們關閉月曆視窗時，出現了以下的錯誤訊息視窗。

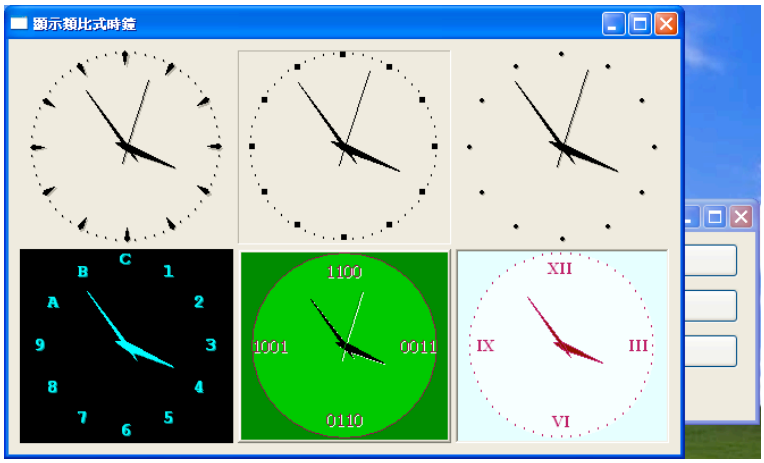


因為我們建立了Calendar模組的TestPanel物件，還有些其他的功能我們沒用到。這裡是說該物件本身有個log屬性，實際上是由run模組中的Log型態所建立的物件，我們沒用到，然後用了個整數-1代進去，因此發生錯誤。還好這個錯誤不是個嚴重的問題，我們關閉「wxPython: stdout/stderr」視窗後程式仍能運作，所以暫時關閉這個錯誤視窗。

至於時鐘的部份，我們將ButtonClick2()方法更改如下。

```
  def ButtonClick2(self, event):
      clockbox = wx.Frame(None, -1, u"顯示類比式時鐘", size=(600, 400))
      panel = AnalogClock.TestPanel(clockbox, -1)
      clockbox.Show(True)
```

由於AnalogClock的TestPanel型態沒有建立新視窗的方法，所以我們先建立一個新視窗，然後在視窗中顯示出類比式時鐘的效果，記得之前註解化的「#self.Bind(...)」要拿掉井字號。由於更改了檔案內容，所以原先執行的程式要先結束（關閉視窗），然後重新執行，結果如下。



最後是電子鐘的部份，我們將ButtonClick3()方法更改如下。

```
def ButtonClick3(self, event):
    clockbox = wx.Frame(None, -1, u"顯示電子式時鐘", size=(340, 300))
    panel = LEDNumberCtrl.TestPanel(clockbox, -1)
    clockbox.Show(True)
```

LEDNumberCtrl模組的TestPanel型態也沒有單獨顯示視窗的方法，所以我們這裡先建立一個視窗，然後把TestPanel放到新的視窗內，結果如下。

當然，別忘了要重新執行，之前註解化的程式碼也要移除註解。

我們把所有程式碼摘錄如下。

```
#-*- coding: UTF-8 -*-

import wx

from demo import Calendar, AnalogClock, LEDNumberCtrl

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(400, 200))
        panel = wx.Panel(self, -1)

        font = wx.SystemSettings_GetFont(wx.SYS_SYSTEM_FONT)
        font.SetPointSize(12)

        vbox = wx.BoxSizer(wx.VERTICAL)

        vbox.Add((-1, 10))
```

```
        hbox1 = wx.BoxSizer(wx.HORIZONTAL)
        self.button1 = wx.Button(panel, 1, u"月曆", size=(365, 30))
        self.Bind(wx.EVT_BUTTON, self. ButtonClick1, id = 1)
        hbox1.Add(self.button1, 0)

        vbox.Add(hbox1, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
        vbox.Add((-1, 10))

        hbox2 = wx.BoxSizer(wx.HORIZONTAL)
        self.button2 = wx.Button(panel, 2, u"時鐘", size=(365, 30))
        self.Bind(wx.EVT_BUTTON, self.ButtonClick2, id = 2)
        hbox2.Add(self.button2, 0)

        vbox.Add(hbox2, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
        vbox.Add((-1, 10))

        hbox3 = wx.BoxSizer(wx.HORIZONTAL)
        self.button3 = wx.Button(panel, 3, u"電子鐘", size=(365, 30))
        self.Bind(wx.EVT_BUTTON, self.ButtonClick3, id = 3)
        hbox3.Add(self.button3, 0)

        vbox.Add(hbox3, 0, wx.EXPAND | wx.LEFT | wx.RIGHT, 12)
        vbox.Add((-1, 10))

        panel.SetSizer(vbox)
        self.Centre()
        self.Show(True)

    def ButtonClick1(self, event):
        calendar = Calendar.TestPanel(self, -1 ,self).TestDlg(event)

    def ButtonClick2(self, event):
        clockbox = wx.Frame(None, -1, u"顯示類比式時鐘", size=(600, 400))
        panel = AnalogClock.TestPanel(clockbox, -1)
        clockbox.Show(True)

    def ButtonClick3(self, event):
        clockbox = wx.Frame(None, -1, u"顯示電子式時鐘", size=(340, 300))
        panel = LEDNumberCtrl.TestPanel(clockbox, -1)
        clockbox.Show(True)

if __name__ == "__main__":
    app = wx.App()
    frame = MyFrame(None, -1, title=u"顯示月曆與時間的小工具")
    app.MainLoop()
```

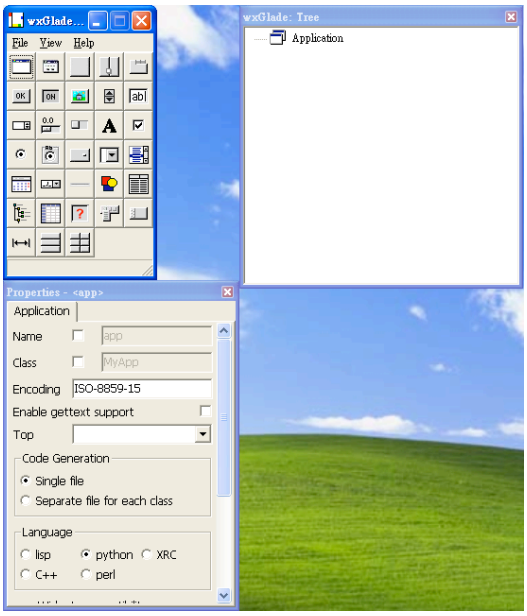
Note

那，我能不能更改Calendar、AnalogClock、LEDNumberCtrl三個模組的內容，使月曆及類比式與電子式時鐘都能依照我希望的外觀顯示呢？如果是仿照範例的寫法，自行發展新的程式，答案當然是可以的，不過若是直接沿用，比如從demo取出的模組，然後依需要修改檔案，這與自行發展新的程式就不一Application樣了。我們若是直接沿用，需要遵守LGPL授權的規範Application，因此當檔案修改完成後，同樣要以LGPL授權的方式發佈，以及釋出原始程式碼。

輔助工具：wxGlade

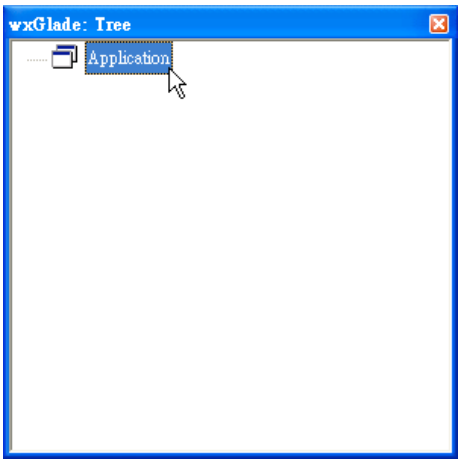
除了完全由自己撰寫wxPython的程式碼外，我們也可以利用由wxPython延伸的**GUI建造器**來產生程式碼。這一類的工具很多，開放原始碼或商業化的軟體都有，我們打算在這裡介紹開放原始碼軟體中的wxGlade，其以MIT License授權。

我們從官網下載安裝檔，然後依安裝指示進行，完成後開啟wxGlade軟體會出現三個視窗，分別是wxGlade主視窗、與Properties視窗。wxGlade主視窗提供建立視窗各種元件的快速按鈕，Tree視窗顯示所建立所有視窗元件（每個物件）的樹狀結構，Properties視窗則可快速設定各項屬性，如右圖。

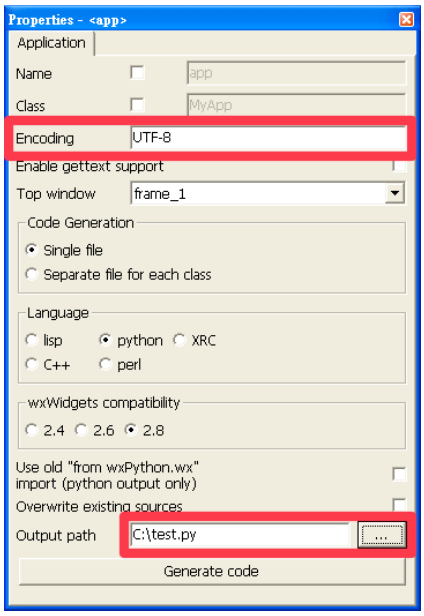


我們以這一章稍早的程式為例，說明如何利用wxGlade快速建立三個按鈕的視窗。

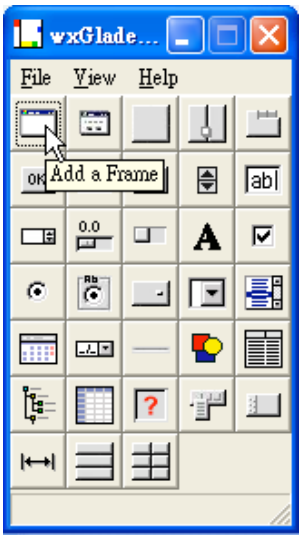
這裡，我們若是點選Tree視窗中的元件，Properties視窗就會出現相對應的屬性值設定。雖然Tree視窗目前只有Application，記不記得我們之前在執行程式的地方建立的app物件，Application就是那個app物件了，如下圖。



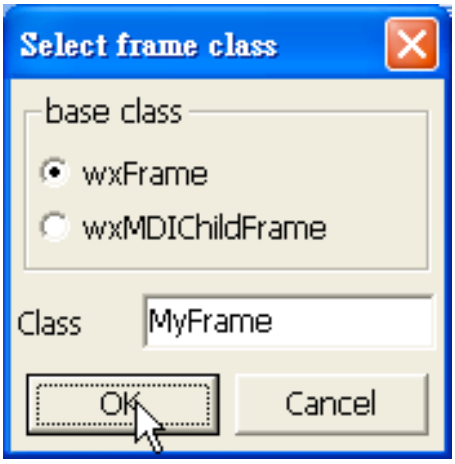
我們把目光轉移到Application的Ptoperties視窗，裡頭有許多欄位可以設定，大部分有預設值，所以我們暫時不用理會，倒是要將Encoding欄從原本的ISO-8859-15改成UTF-8，這是為了等一下輸入中文之用，另外，底下的Output path欄填入C:\test.py，其作為我們最後輸出檔案的路徑與檔名，如右。



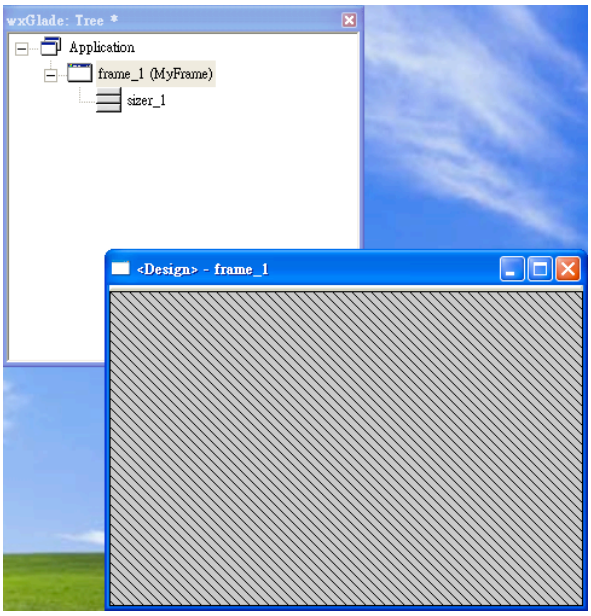
接著，我們點擊wxGlade視窗中建立Frame的按鈕，如下。



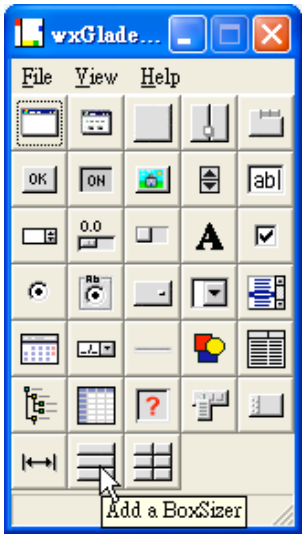
然後出現Select frame class的視窗，這裡可以選擇所要建立的Frame要繼承自wxFrame或wxMDIChildFrame，以及所建立型態（class）的名稱，其實都有預設值，因此我們直接點擊OK即可。



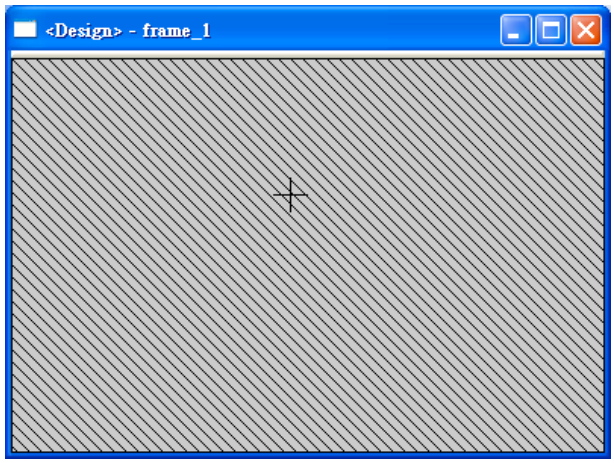
點擊OK之後，我們可以看到Tree視窗出現了frame_1的物件，也有一個<Design> - frame_1的新視窗，這個視窗是wxGlade所見即所得的預覽結果。



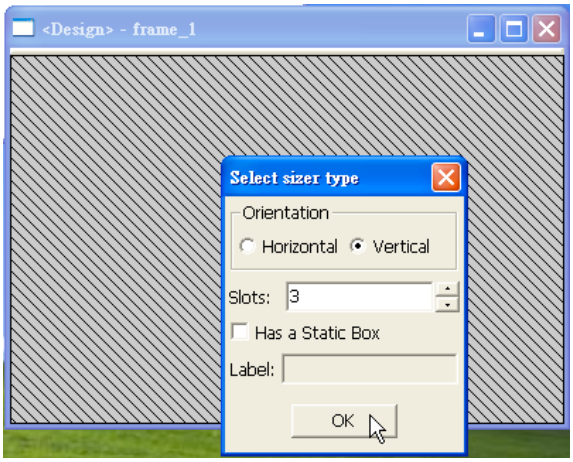
我們回到wxGlade視窗，點擊建立BoxSizer的按鈕，滑鼠游標就會變成十字形。



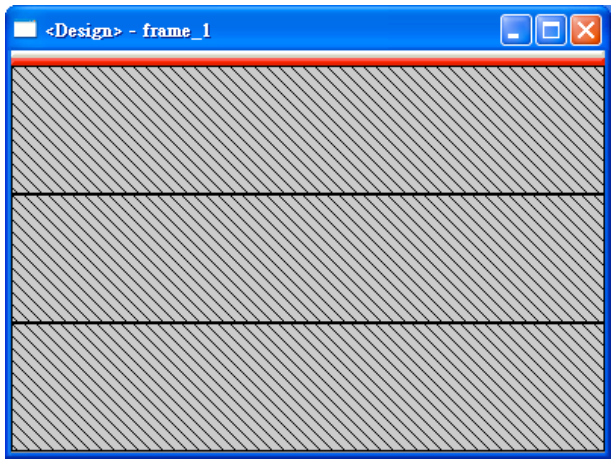
然後將滑鼠游標移到<Design> - frame_1視窗上。



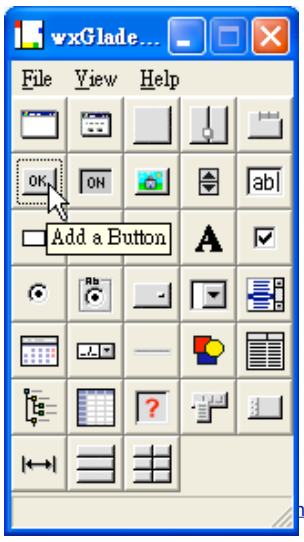
點一下，接著出現Select sizer type的視窗，這裡，我們在Orientation欄選擇Vertical（垂直排列），底下的Slots欄有調整為3，因為有三個按鈕。完成輸入後，點擊OK。



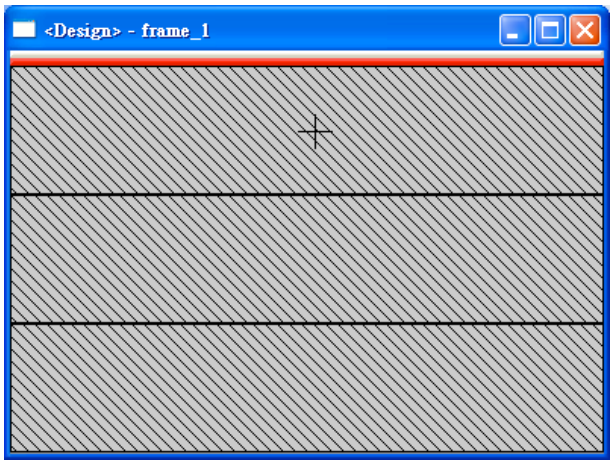
點擊OK後，我們可以看到<Design> - frame_1視窗從上到下分成三欄，中間用兩條黑線相隔。



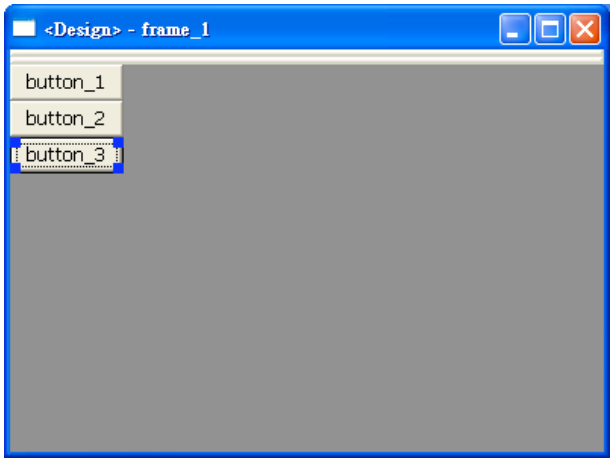
然後我們將目光移回wxGlade視窗，找到建立Button的按鈕，點擊後將滑鼠游標移回<Design> - frame_1視窗。



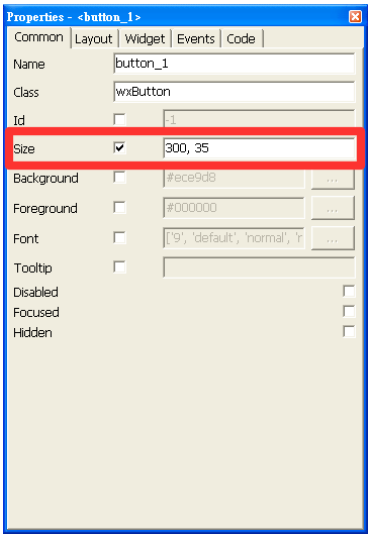
滑鼠游標仍是變為十字形。



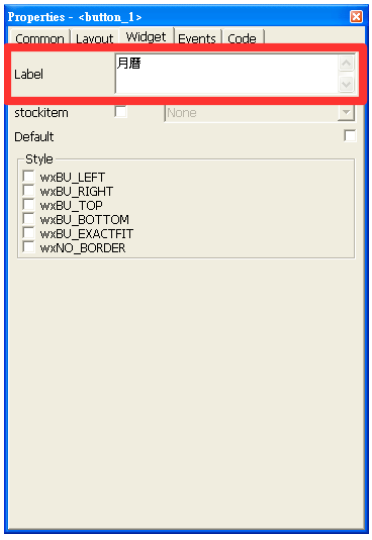
我們用滑鼠在第一欄點一下，再去點擊建立Burron的按鈕，然後移回到第二欄點一下，同樣的動作再做一次，如此，<Design> - frame_1視窗就出現了三個新按鈕。



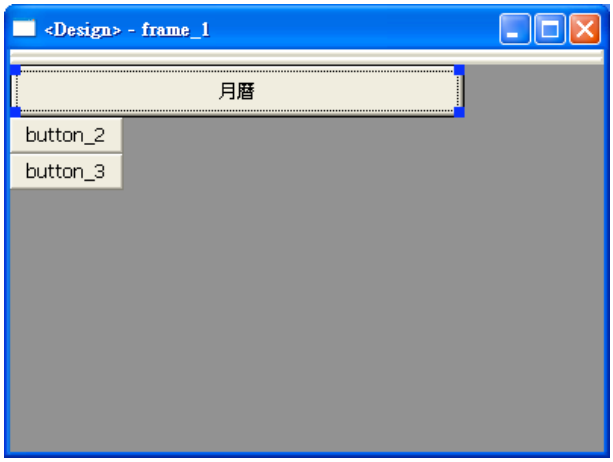
然後我們利用滑鼠選取第一個按鈕，將目光移到Properties視窗，所有Button的屬性出現在這，然後我們將Size欄勾選，把預設值-1, -1改為300, 35，這是說我們將按鈕大小調整為300×35的大小。



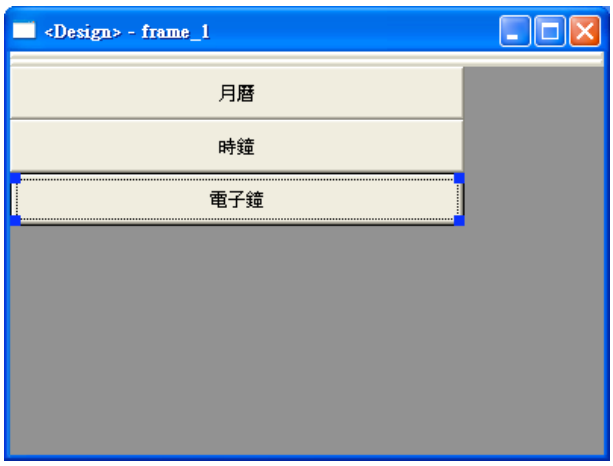
然後換到Widget分頁，我們在此將Label欄由button_1更改為月曆，這會是按鈕所呈現的文字。



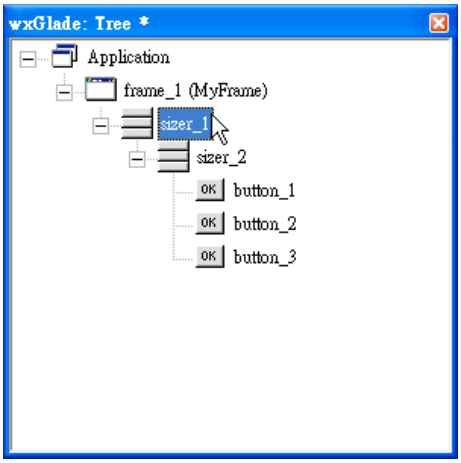
我們可以看到<Design> - frame_1視窗中的第一個按鈕，如我們做過的調整而改變了。



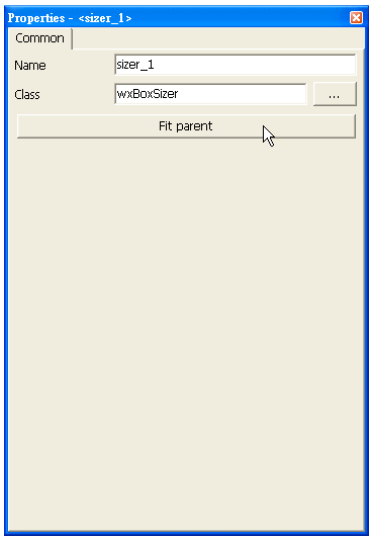
當然，我們要完成其他兩個按鈕的變更。



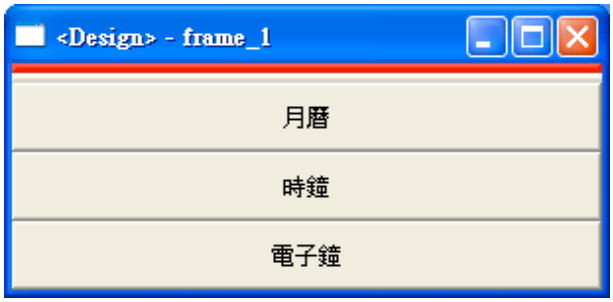
然後我們要將三個按鈕符合視窗的大小，於是，我們把滑鼠游標移到Tree視窗，點選sizer_1物件。



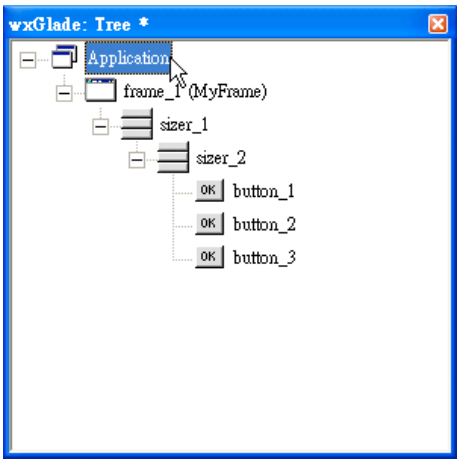
目光回到Properties視窗，這自然會是sizer_1屬性質的設定，但是除了名稱外，倒是只多了個Fit parent按鈕，沒錯，我們就是要點擊這個按鈕。



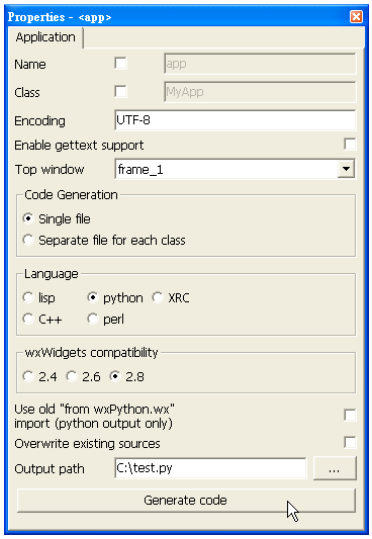
然後，我們可以看到<Design> - frame_1視窗中，三個按鈕確實緊緊扎在視窗中了。



我們很快的把三個按鈕完成了，接下來便是產生程式碼。我們將目光移到Tree視窗，點選Application物件。



然後再將目光移回Properties視窗，當然，就是點擊最下面的Generate code按鈕。



稍後出現一個Information視窗，告訴我們程式碼成功的產出了。

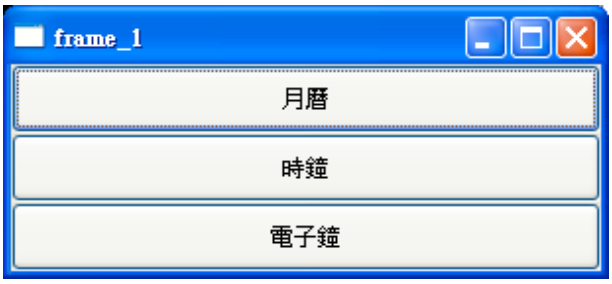
這些程式碼長得怎麼樣呢？如下。

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# generated by wxGlade 0.6.3 on Sun Jan 18 16:02:07 2009

import wx

# begin wxGlade: extracode
# end wxGlade



class MyFrame(wx.Frame):
    def __init__(self, *args, **kwds):
        # begin wxGlade: MyFrame.__init__
        kwds["style"] = wx.DEFAULT_FRAME_STYLE
        wx.Frame.__init__(self, *args, **kwds)
        self.button_1 = wx.Button(self, -1, u"月曆")
        self.button_2 = wx.Button(self, -1, u"時鐘")
        self.button_3 = wx.Button(self, -1, u"電子鐘")

        self.__set_properties()
        self.__do_layout()
        # end wxGlade

    def __set_properties(self):
        # begin wxGlade: MyFrame.__set_properties
        self.SetTitle("frame_1")
        self.button_1.SetMinSize((300, 35))
        self.button_2.SetMinSize((300, 35))
        self.button_3.SetMinSize((300, 35))
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: MyFrame.__do_layout
        sizer_1 = wx.BoxSizer(wx.VERTICAL)
        sizer_2 = wx.BoxSizer(wx.VERTICAL)
        sizer_2.Add(self.button_1, 0, 0, 0)
        sizer_2.Add(self.button_2, 0, 0, 0)
        sizer_2.Add(self.button_3, 0, 0, 0)
        sizer_1.Add(sizer_2, 1, wx.EXPAND, 0)
        self.SetSizer(sizer_1)
        sizer_1.Fit(self)
        self.Layout()
        # end wxGlade

# end of class MyFrame


if __name__ == "__main__":
```

```
app = wx.PySimpleApp(0)
wx.InitAllImageHandlers()
frame_1 = MyFrame(None, -1, "")
app.SetTopWindow(frame_1)
frame_1.Show()
app.MainLoop()
```

裡頭有很多東西看不懂？別擔心，我們在這裡只是提供Python語言的入門，以及程式設計的概念，當各位看官繼續深入Python語言的境地，許多看不懂東西就會慢慢的自然而然的心神領會。

還是來看看執行這個程式的結果，如下。



還不錯呢？對不對啊？

下一步

wxPyWiki提供的許多有用的資源，包含提供如何學習wxPython的參考建議 -- How to learn wxPython.，也提供了部份線上教材，而官網所建議的wxPython in Action是一本很好的入門書，也是翻查wxPython資料便捷的工具書。

當然，還有很多東西我們沒有提，然而作為入門的介紹，我們認為對於wxPython -- 應用程式介面的設計，這些材料即可。下一篇開始，我們轉換焦點，來看看如何利用Python快速建立網站。