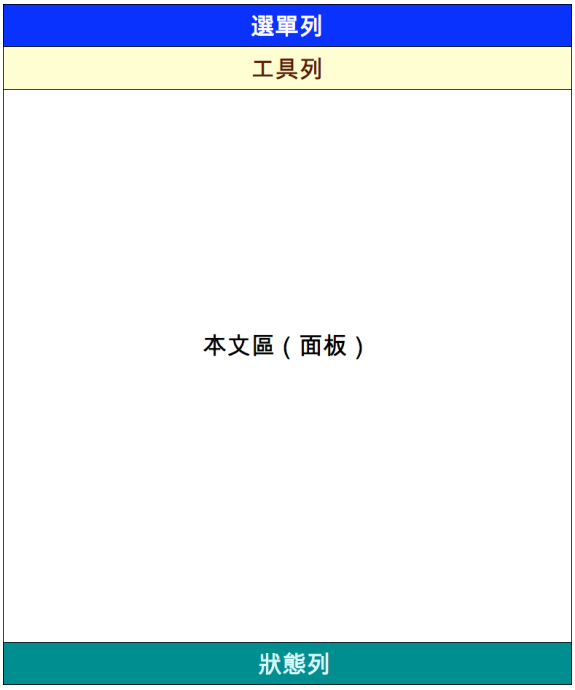
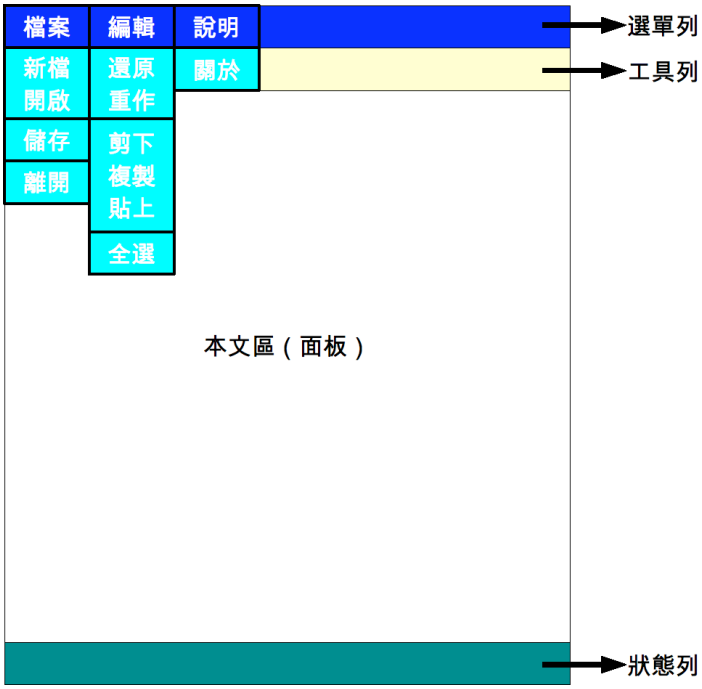
第十四章 簡單的文字編輯器

上一章中，我們看到如何在面板中排列提示訊息、輸入文字方塊，以及顯示歡迎及現在時間的訊息。這一章裡，我們以簡單的文字應用軟體為例，說明其他常見的視窗元件，包含選單列、工具列、狀態列與對話視窗如何建立。

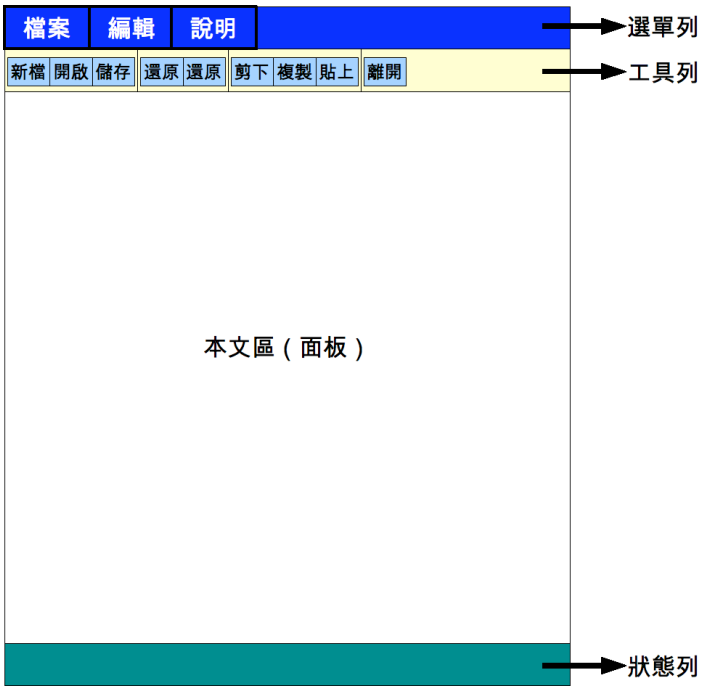
圖形使用者介面基本的規劃如下圖。



第一行的選單列，預計有以下三個功能表。



下一行的工具列，利用圖形將常用功能製作成九個按鈕。



視窗下方的狀態列則作為提供提示訊息。

建立視窗及狀態列

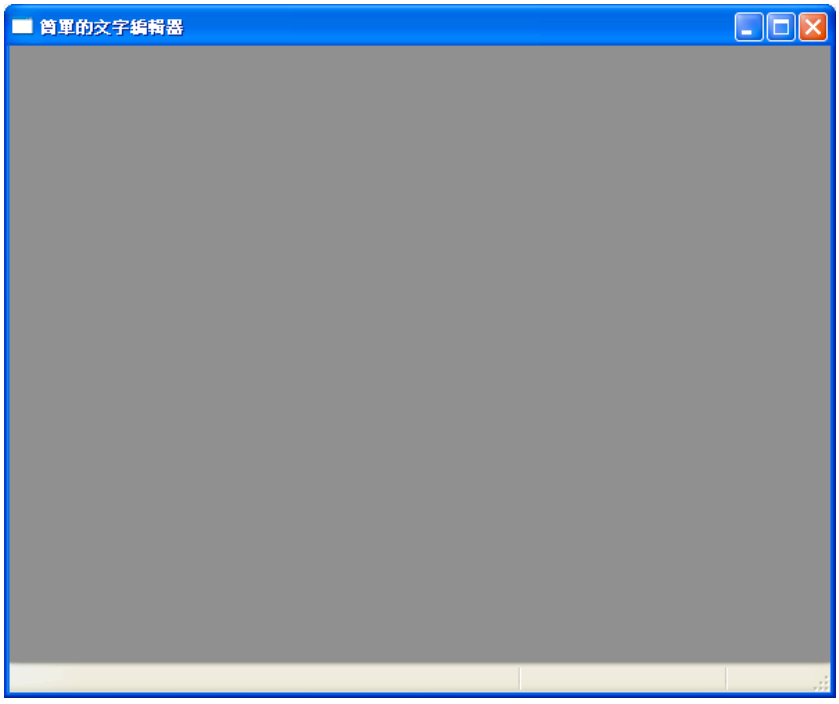
以下的程式建立一個具有狀態列的視窗。

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

import wx

class SEditor(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(600, 500))

        # 使用statusbar()方法建立狀態列
        self.statusBar()

        # 視窗顯示的設定
        self.Show(True)

    def statusBar(self):
        self.statusbar = self.CreatStatusBar()
        self.statusbar.SetFieldsCount(3)
        self.statusbar.SetStatusWidths([-5, -2, -1])

if __name__ == "__main__":
    app = wx.App()
    frame = SEditor(None, -1, u"簡單的文字編輯器")
    app.MainLoop()
```

視窗大小設定成600×500，標題列提供的是「簡單的文字編輯器」的u前綴字串。然後我們利用自訂的方法statusBar()設定狀態列，在statusBar()方法的定義裡，首先將屬性statusbar用wx.Frame型態的CreateStatusBar()方法建立空白單欄的狀態列，這時statusbar的型態為

wx.ToolBar，再來便是利用SetFieldsCount()將其劃分為三個欄位，最後用wx.ToolBar型態的SetStatusWidths()方法依比例分配欄位長度。

要注意SetStatusWidths()方法的參數為一個串列，其中為幾個負整數組成，這是依比例分配欄位長度的方式。由於還沒有提供狀態列會顯示的訊息，所以狀態列暫時為空白，視窗也以系統預設的顏色顯示。執行結果如下。



我們可以發現建立狀態列時，並沒有用到Sizer，狀態列直接排列在視窗的底部，這是由於wxPython對於某些視窗元件有預設的排列方式，或稱wxPython具有內建的Sizer，包含狀態列以及選單列、工具列等。接下來我們以這個程式為骨架，利用內建的Sizer逐一擴展完成規劃的最終版本。

另還有一點要注意的是名稱statusBar，第一個字母為小寫的s，這是因為wxPython內建的名稱如Show、CreateStatusBar、SetFieldsCount等，多半是幾個英文單字連結在一起，每個單字的第一個字母為大寫。由於wxPython裡頭定義了許多名稱，為了避免名稱衝突，所以我們將第一個單字的字母設為小寫。

也基於這個理由，我們將所有的屬性名稱，全部單字都採英文小寫的字母。

建立本文區

要建立文字輸入的本文區很簡單，我們利用自定義的textControl()方法。

```
# 使用self.textControl建立本文區
self.textControl()
```

textControl()方法的定義如下。

```
def textControl(self):
    self.text = wx.TextCtrl(self, ID_TEXT, style=wx.TE_MULTILINE)
```

其中有個ID_TEXT變數，這不是wxPython裡頭定義的變數，而是我們在定義SEditor型態之前，先要建立這個全域變數。

```
ID_TEXT = 101
```

我們在上一章中已經提過TextCtrl型態，不過那時候單純的提供兩個參數，分別是panel及-1，而建立wx.TextCtrl型態的物件的時候，實際需要的參數如下。

```
wx.TextCtrl(parent, id, value="", pos=wx.DefaultPosition,
size=wx.DefaultSize, style=0, validator=wx.DefaultValidator,
name=wx.TextCtrlNameStr)
```

其中只有parent及id是必要的，其他的參數都有預設值，這裡我們給style提供wx.TE_MULTILINE，允許使用者在文字控制方塊輸入多行。

重新執行程式，會得到如下的結果。

當style的標籤設定為wx.TE_MULTILINE的時候，視窗元件TextCtrl型態就會自動擴展充滿整個視窗，同時視窗右邊的捲軸也會自動出現，依照作業系統預定的風格展現出來。這也是wxPython的特色之一，利用wxPython設計應用程式介面，其介面會呈現出該平台預設的樣貌。

建立工具列

我們自行定義方法來建立工具列。

```
# 使用toolBar()方法建立工具列
self.toolBar()
```

同樣的，我們要在SEditor型態定義前加入一些ID變數的設定。

```
ID_NEW = 801
ID_OPEN = 802
ID_SAVE = 803
ID_UNDO = 805
ID_REDO = 806
ID_CUT = 807
ID_COPY = 808
ID_PASTE = 809
ID_EXIT = 811
ID_ABOUT = 813
```

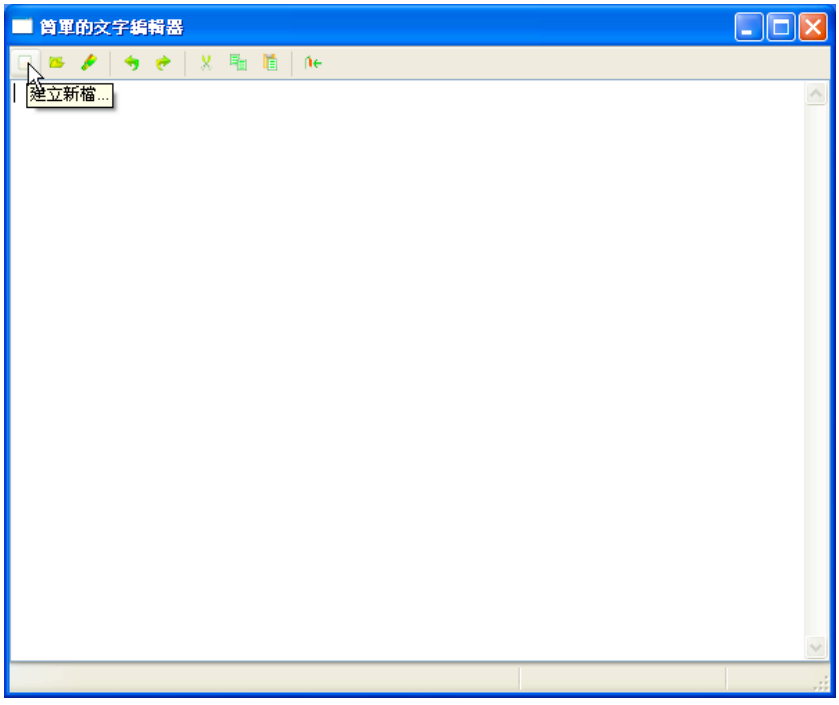
方法toolBar()的定義如下。

```
def toolBar(self):
    self.toolbar = self.CreateToolBar()
```

```
        self.toolbar.AddSimpleTool(ID_NEW, wx.Bitmap("icons/filenew.png"), \
                                                              u"建立新檔...")
        self.toolbar.AddSimpleTool(ID_OPEN, wx.Bitmap("icons/folder.png"), \
                                                              u"開啟舊檔...")
        self.toolbar.AddSimpleTool(ID_SAVE, wx.Bitmap("icons/filesave.png"),\
                                                              u"儲存檔案...")
        self.toolbar.AddSeparator()

        self.toolbar.AddSimpleTool(wx.ID_UNDO, wx.Bitmap("icons/undo.png"), \
                                                        u"還原...", ""u"還原...")
        self.toolbar.AddSimpleTool(wx.ID_REDO, wx.Bitmap("icons/redo.png"), \
                                                        u"重作...", ""u"重作...")
        self.toolbar.AddSeparator()


        self.toolbar.AddSimpleTool(ID_CUT, wx.Bitmap("icons/editcut.png"), \
                                                                u"剪下...")
        self.toolbar.AddSimpleTool(ID_COPY, wx.Bitmap("icons/editcopy.png"), \
                                                                u"複製...")
        self.toolbar.AddSimpleTool(ID_PASTE, wx.Bitmap("icons/editpaste.png"),\
                                                                u"貼上...")
        self.toolbar.AddSeparator()

        self.toolbar.AddSimpleTool(ID_EXIT, wx.Bitmap("icons/exit.png"), \
                                                            u"關閉文字編輯器")
        self.toolbar.Realize()
```

由wx.Frame的CreateToolBar()方法建立工具列之後，接著儲存到屬性toolbar之中，這時toolbar的型態便是wx.ToolBar，然後用AddSimpleTool()方法加入新的工具圖示及輔助說明文字。當我們要把工具列的按鈕分組時，AddSeparator()方法可以畫出分隔線。最後，要用Realize()方法讓載入的圖形檔案得以顯示。

呼叫CreateToolBar()時沒有給予參數，全部以預設值載入。而AddSimpleTool()中我們給予三個參數，分別是ID值、載入的圖檔與輔助說明字串，ID值儲存在ID變數之中，wx.Bitmap()型態轉換圖檔成為可以在螢幕上顯示的格式，移動滑鼠游標到用作工具列按鈕的圖形上時，便會顯示輔助說明字串，如下。



我們會為每個工具列的按鈕寫方法，並且讓方法與按鈕產生連結，使按下按鈕發揮工具的效用，不過稍待一會，因為工具列的按鈕是從選單中選取出來的，所以，我們先來建立選單列，之後再把工具列與選單列的物件連結到產生功能的方法。

建立選單列

我們仍以自行定義的方法來建立選單列。

```
# 使用menuBar()方法建立選單列
self.menuBar()
```

我們有個ID變數還沒設定。

```
ID_ALL = 810
```

方法menuBar()定義如下。

```
def menuBar(self):
    menubar = wx.MenuBar()

    file = wx.Menu() # 建立檔案選單
    new = wx.MenuItem(file, ID_NEW, u"&新檔", u"建立新檔...")
    file.AppendItem(new)

    open = wx.MenuItem(file, ID_OPEN, u"&開啟", u"開啟舊檔...")
    file.AppendItem(open)
    file.AppendSeparator()

    save = wx.MenuItem(file, ID_SAVE, u"&儲存", u"儲存檔案...")
    file.AppendItem(save)
    file.AppendSeparator()

    quit = wx.MenuItem(file, ID_EXIT, u"&離開", u"離開文字編輯器...")
    file.AppendItem(quit)

    menubar.Append(file, u"&檔案") # 將檔案選單加入選單列

    edit = wx.Menu() # 建立編輯選單
    undo = wx.MenuItem(edit, wx.ID_UNDO, u"&還原", u"還原...")
    edit.AppendItem(undo)

    redo = wx.MenuItem(edit, wx.ID_REDO, u"&重作", u"重作...")
    edit.AppendItem(redo)
    edit.AppendSeparator()

    cut = wx.MenuItem(edit, ID_CUT, u"&剪下", u"剪下...")
    edit.AppendItem(cut)

    copy = wx.MenuItem(edit, ID_COPY, u"&複製", u"複製...")
    edit.AppendItem(copy)

    paste = wx.MenuItem(edit, ID_PASTE, u"&貼上", u"貼上...")
    edit.AppendItem(paste)
    edit.AppendSeparator()

    all = wx.MenuItem(edit, ID_ALL, u"&全選", u"全部選取...")
    edit.AppendItem(all)

    menubar.Append(edit, u"&編輯") # 將編輯選單加入選單列
```

```
        help = wx.Menu() # 建立說明選單
        about = wx.MenuItem(help, ID_ABOUT, u"&關於", \
                                                u"顯示文字編輯器的版本資訊...")
        help.AppendItem(about)

        menubar.Append(help, u"&說明") # 將說明選單加入選單列

        self.SetMenuBar(menubar)
```

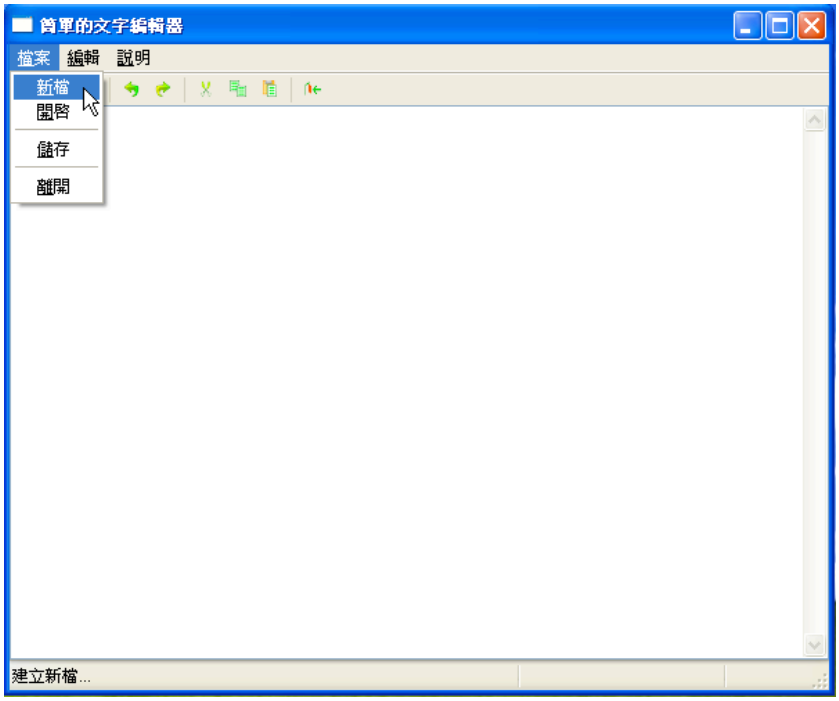
首先將wx.MenuBar建立到變數menubar之中，這就是我們所要建立的選單列，接著我們依次建立檔案、編輯與說明三個選單，分別由file、edit、help三個變數來著手，wx.Menu即是選單的物件，其為wx.MenuBar中的個別元素。

因此，選項建立完成後隨即利用wx.MenuBar的Append()方法，將file、edit、help三個wx.Menu物件附加到menubar之中。我們先來看看選項的建立。

```
new = wx.MenuItem(file, ID_NEW, u"&新檔", u"建立新檔...")
```

選項是利用wx.MenuItem物件，共需四個物件，file為所繼承的型態，ID_NEW為ID變數，接著兩個u前綴字串，第一個為選單中顯示的名稱，第二個則是出現在狀態列中的提示訊息。

在檔案選單中，利用new、open、save、quit四個變數建立wx.MenuItem物件，然後利用wx.Menu型態的AppendItem()方法附加到選單中，而AppendSeparator()方法為附加分隔線。由此，我們可以看到所建立出選單的結果。



建立新檔的訊息視窗

開啟一個應用程式通常會直接載入預設的檔名，常見的如未命名、文件1、無標題1等，在這個程式中我們不打算這麼做，反倒是在程式開始執行時，先跳出一個輸入檔名的訊息視窗，讓使用者輸入所要建立新檔的檔名。當然，我們仍是可以先載入預設的檔名

因此，在初始化方法中，我們以NewName()方法取得新的檔名。

```
# 以NewName()方法取得檔案名稱
self.NewName()
```

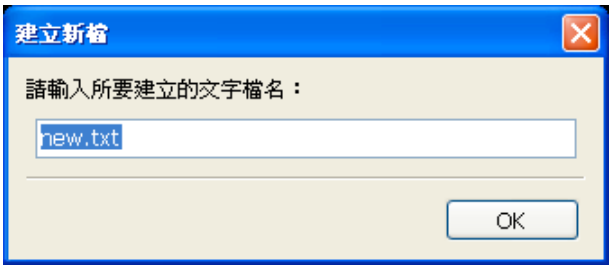
NewName()方法定義如下。

```
    def NewName(self):
        dialog = wx.TextEntryDialog(None, u"請輸入所要建立的文字檔名：",\
                                          u"建立新檔", "new.txt", style=wx.OK)
        if dialog.ShowModal() == wx.ID_OK:
            self.filename = dialog.GetValue()
        dialog.Destroy()
```

wx.TextEntryDialog為要求使用者輸入文字的對話視窗，有五個參數，第一個為所要繼承的型態，這裡設為None，第二個為視窗中的提示訊息，第三個則是視窗的標題列，第四個為我們打算載入的預設名稱，最後一個參數style設為wx.OK，使視窗會產生一個【OK】的按鈕。

wx.TextEntryDialog的ShowModal()方法會回傳某些特定的常數值，如wx.ID_OK或是wx.ID_CANCEL等，其用途為儲存使用者所按下的按鈕種類，因而這裡直接用作條件判斷，假如使用者按下【OK】，屬性filename就會從wx.TextEntryDialog取得數值。

執行結果如下。



最後一行呼叫Destroy()方法，這是為了結束對話視窗，不然程式留在記憶體中揮之不去，除非用其他強制手段結束執行。當然，按下【OK】後，就會進入文字編輯器的視窗。

開始編輯後的狀態列提示

在建立狀態列的時候，狀態列就已經分為三欄，其中最左邊的一欄顯示選單操作的提示訊息，我們現在替其他兩欄撰寫開始編輯後的狀態列提示，也就是使用者在本文區鍵入資料後，狀態列就會出現相對應的提示訊息。

中間那欄我們打算簡單的顯示「編輯本文」的訊息，而右邊那欄則顯示目前鍵入的字元數，這些我們另外寫一個方法進行處理。

```
    def OnTextChanged(self, event):
        self.statusbar.SetStatusText(u"編輯本文...", 1)

        self.lines = self.text.GetLastPosition()
        self.statusbar.SetStatusText(unicode(self.lines), 2)

        event.Skip()
```

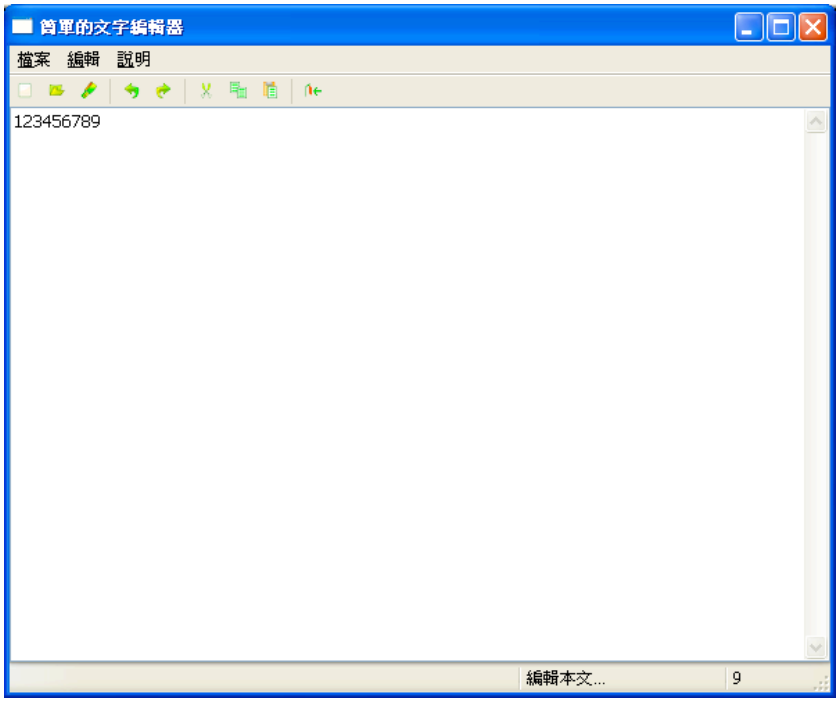
假如OnTextChanged()方法被呼叫，statusbar屬性就以wx.StatusBar的SetStatusText()方法設定顯示的提示訊息，其中，中間那欄直接顯示「編輯本文...」的訊息，右邊那欄則先由lines屬性從text屬性以wx.TextCtrl的GetLastPosition()取得最後輸入的位置座標，就會顯示出目前輸入的字元數。

由於wx.TextCtrl的控制方塊對於輸入的文字，全部都被當成一行來處理，如果中間有按下Enter鍵，那如同鍵入“\n”的跳脫序列，所以wx.TextCtrl的座標位置處理，好比記錄共輸入了多少字元數，包括跳脫序列。最後wx.Event型態的參數event呼叫Skip()方法，這是作為處理事件之用。

這裡OnTextChanged的名稱亦作為區別wxPython的內建名稱，因為這部份我們所用的方法名稱沒有以On開頭的。同樣的，我們需要把OnTextChanged()與程式事件處理連結，於是textControl()方法的定義要加入以下的程式碼。

```
    # 功能連結
    self.text.Bind(wx.EVT_TEXT, self.OnTextChanged, id=ID_TEXT)
```

重新執行程式，會得到像是如下的結果。

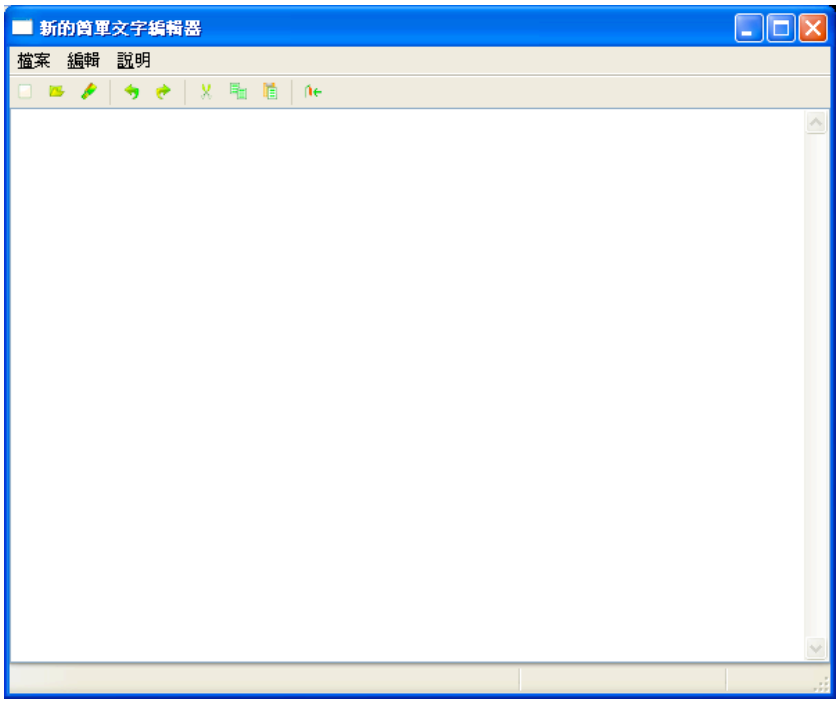


檔案選單及警告訊息

接下來我們開始撰寫選單列及工具列各個指令的方法，先來看到檔案選單，包括工具列中「新檔」、「開啟」、「儲存」三個按鈕。「新檔」方面，我們以NewApplication()的方法來處理。

```
    def NewApplication(self, event):
        new = SEditor(None, -1, u"新的簡單文字編輯器")
```

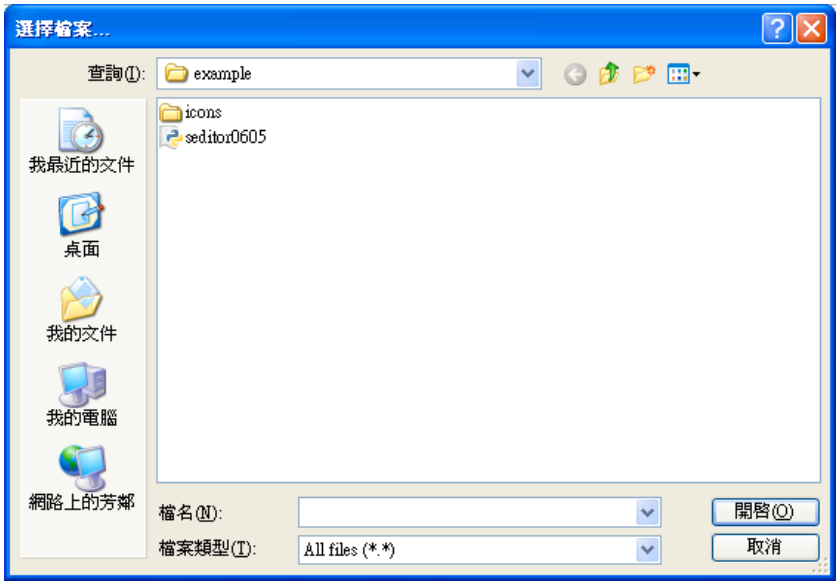
建立新檔如同我們再次執行程式，不過這裡把原先的標題改成了「新的簡單文字編輯器」。



「開啟」方面，我們建立系統預設的開啟檔案視窗。

```
  def OpenFile(self, event):
      self.dirname = os.getcwd()
      file_dialog = wx.FileDialog(self, u"選擇檔案...", self.dirname,\
                                                        style=wx.OPEN)
      if file_dialog.ShowModal() == wx.ID_OK:
          self.filename = file_dialog.GetFilename()
          self.dirname = file_dialog.GetDirectory()
          f = open(os.path.join(self.dirname, self.filename), "r")
          self.text.SetValue(f.read())
          f.close()
      file_dialog.Destroy()
```

這裡先以dirname屬性用os模組的getcwd()函數取得目前所在的路徑目錄，然後建立wx.FileDialog的對話視窗，如下。



wx.FileDialog用了四個參數，第一個是所繼承的參數，第二個則是給標題列的字串，第三個則是開啟的路徑目錄，第四個style則是設定為wx.OPEN。當開啟檔案的對話視窗建立後，屬性filename抓取使用者所選的檔案名稱，dirname則重新取得改變後的路徑目錄。

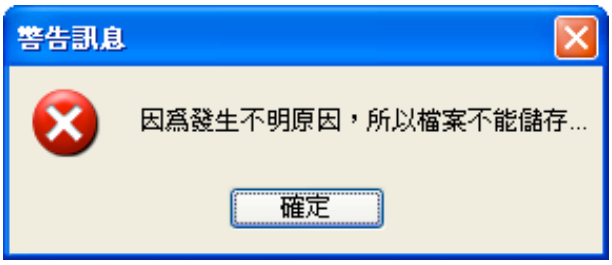
接著以變數f用內建函數open()以“r”模式開啟檔案，然後text屬性以wx.TextCtrl的SetValue()方法載入檔案內容到text之中，如此一來，開啟舊檔的工作就完成了。

「儲存」所用的方法也極為類似，然而，用作檔名的filename如果是空字串，就會發生問題。

```
  def SaveFile(self, event):
      if self.filename:
          f = open(self.filename, "w")
          text = self.text.GetValue()
          f.write(text)
          f.close()
      else:
          alert_dialog = wx.MessageDialog(None, \
                                      u"因為發生不明原因，所以檔案不能儲存...", \
                                      u"警告訊息", wx.OK | wx.ICON_ERROR)
          if alert_dialog.ShowModal() == wx.ID_OK:
              alert_dialog.Destroy()
```

SaveFile()方法的定義之中，首先判斷filename是否為空字串，若filename不為空字串，檔案就以變數f用內建函數open()以“w”模式開啟，然後將text屬性的內容以wx.TextCtrl的GetValue方法取得，接著寫入到變數f之中，如此就完成了儲存的工作。

如果filename為空字串，程式就會跳到else陳述執行，這裡以wx.MessageDialog建立一個提供警告訊息的視窗，如下。



雖然發生的原因清楚，不過我們頑皮一點，告訴使用者結果，也就是檔案不能被儲存，卻沒有清楚的告訴使用者發生的原因。wx.MessageDialog所用的參數如同其他建立對話視窗的型態，第四個參數中的wx.ICON_ERROR是讓視窗中出現系統預設的警示圖像。

檔案選單還有一個「離開」，這也是工具列中最右邊的按鈕。

```
  def QuitApplication(self, event):
      self.Close(True)
```

利用wx.Frame的Close()方法，就能夠結束程式的執行。最後，當然要在menuBar()與toolBar()的定義中加入功能連結。

以下是menuBar()的部份。

```
  self.Bind(wx.EVT_MENU, self.NewApplication, id=ID_NEW)
  self.Bind(wx.EVT_MENU, self.OpenFile, id=ID_OPEN)
  self.Bind(wx.EVT_MENU, self.SaveFile, id=ID_SAVE)
  self.Bind(wx.EVT_MENU, self.QuitApplication, id=ID_EXIT)
```

以下則是toolBar()的部份。

```
  self.Bind(wx.EVT_TOOL, self.NewApplication, id=ID_NEW)
  self.Bind(wx.EVT_TOOL, self.OpenFile, id=ID_OPEN)
```

```
self.Bind(wx.EVT_TOOL, self.SaveFile, id=ID_SAVE)
self.Bind(wx.EVT_TOOL, self.QuitApplication, id=ID_EXIT)
```

編輯選單的功能

編輯選單中有「還原」、「重作」、「剪下」、「複製」、「貼上」、「全選」等選項，工具列則包含除了「全選」之外的五個。其實這些方法在wx.TextCtrl中都有了，因此我們可以直接套用。

```
def OnUndo(self, event):
    self.text.Undo()

def OnRedo(self, event):
    self.text.Redo()

def OnCut(self, event):
    self.text.Cut()

def OnCopy(self, event):
    self.text.Copy()

def OnPaste(self, event):
    self.text.Paste()

def OnAll(self, event):
    self.text.SelectAll()
```

我們仍以自行定義的方法套用，這個好處是將來如果要擴展功能，直接從這些方法定義的地方著手就可以了。同樣的，功能連結不要忘了加進去。

以下是menuBar()的部份。

```
self.Bind(wx.EVT_MENU, self.OnUndo, id=wx.ID_UNDO)
self.Bind(wx.EVT_MENU, self.OnRedo, id=wx.ID_REDO)
self.Bind(wx.EVT_MENU, self.OnCut, id=ID_CUT)
self.Bind(wx.EVT_MENU, self.OnCopy, id=ID_COPY)
self.Bind(wx.EVT_MENU, self.OnPaste, id=ID_PASTE)
self.Bind(wx.EVT_MENU, self.OnAll, id=ID_ALL)
```

以下則是toolBar()的部份。

```
self.Bind(wx.EVT_TOOL, self.OnUndo, id=wx.ID_UNDO)
self.Bind(wx.EVT_TOOL, self.OnRedo, id=wx.ID_REDO)
self.Bind(wx.EVT_TOOL, self.OnCut, id=ID_CUT)
self.Bind(wx.EVT_TOOL, self.OnCopy, id=ID_COPY)
self.Bind(wx.EVT_TOOL, self.OnPaste, id=ID_PASTE)
```

說明選單中的關於視窗

關於視窗顯示軟體的作者、版本等資訊，我們仍以wx.MessageDialog的方式來建立一個新的顯示說明文字的視窗。首先，我們將所要顯示的文字以三引號字串建立為全域變數。

```
message = u"""

簡單的文字編輯器 v0.06.05


這是在《電腦做什麼事》第十四章中的範例程式。

作　　者：張凱慶
電子信箱：kaichingc@gmail.com
```

```
    歡迎造訪 http://pydoing.blogspot.com/

    Copyright 2008

    """
```

接著寫About()方法產生wx.MessageDialog的對話視窗。

```
    def About(self, event):
        about_dialog = wx.MessageDialog(self, message, u"SEditor 版本資訊",\
                                                                    wx.OK)
        if about_dialog.ShowModal() == wx.ID_OK:
            about_dialog.Destroy()
```

menuBar()仍是要加入的功能連結，不然關於視窗不會出現。

```
    self.Bind(wx.EVT_MENU, self.About, id=ID_ABOUT)
```

顯示如下。

最後我們把所有的程式碼列出如下，供參考。

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

    import wx, os

    message = u"""

    簡單的文字編輯器 v0.06.05


    這是在《電腦做什麼事》第十四章中的範例程式。

    作　　者：張凱慶
    電子信箱：kaichingc@gmail.com

    歡迎造訪 http://pydoing.blogspot.com/

    Copyright 2008

    """
```

```
ID_TEXT = 101

ID_NEW = 801
ID_OPEN = 802
ID_SAVE = 803
ID_UNDO = 805
ID_REDO = 806
ID_CUT = 807
ID_COPY = 808
ID_PASTE = 809
ID_ALL = 810
ID_EXIT = 811
ID_ABOUT = 813

class SEditor(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, size=(600, 500))

        # 兩個在狀態列顯示的屬性初值設定
        self.modify = False
        self.lines = 0

        # 以NewName()方法取得檔案名稱
        self.NewName()

        # 使用menuBar()方法建立選單列
        self.menuBar()

        # 使用toolBar()方法建立工具列
        self.toolBar()

        # 使用self.textControl建立本文區
        self.textControl()

        # 使用statusBar()方法建立狀態列
        self.statusBar()

        # 視窗顯示的設定
        self.Show(True)

    def menuBar(self):
        menubar = wx.MenuBar()

        file = wx.Menu() # 建立檔案選單
        new = wx.MenuItem(file, ID_NEW, u"&新檔", u"建立新檔...")
        file.AppendItem(new)

        open = wx.MenuItem(file, ID_OPEN, u"&開啟", u"開啟舊檔...")
        file.AppendItem(open)
        file.AppendSeparator()

        save = wx.MenuItem(file, ID_SAVE, u"&儲存", u"儲存檔案...")
        file.AppendItem(save)
        file.AppendSeparator()

        quit = wx.MenuItem(file, ID_EXIT, u"&離開", u"離開文字編輯器...")
        file.AppendItem(quit)
```

```
        menubar.Append(file, u"&檔案") # 將檔案選單加入選單列

        edit = wx.Menu() # 建立編輯選單
        undo = wx.MenuItem(edit, wx.ID_UNDO, u"&還原", u"還原...")
        edit.AppendItem(undo)

        redo = wx.MenuItem(edit, wx.ID_REDO, u"&重作", u"重作...")
        edit.AppendItem(redo)
        edit.AppendSeparator()

        cut = wx.MenuItem(edit, ID_CUT, u"&剪下", u"剪下...")
        edit.AppendItem(cut)

        copy = wx.MenuItem(edit, ID_COPY, u"&複製", u"複製...")
        edit.AppendItem(copy)

        paste = wx.MenuItem(edit, ID_PASTE, u"&貼上", u"貼上...")
        edit.AppendItem(paste)
        edit.AppendSeparator()

        all = wx.MenuItem(edit, ID_ALL, u"&全選", u"全部選取...")
        edit.AppendItem(all)

        menubar.Append(edit, u"&編輯") # 將編輯選單加入選單列

        help = wx.Menu() # 建立說明選單
        about = wx.MenuItem(help, ID_ABOUT, u"&關於", \
                                                u"顯示文字編輯器的版本資訊...")
        help.AppendItem(about)

        menubar.Append(help, u"&說明") # 將說明選單加入選單列

        self.SetMenuBar(menubar)

        # 功能連結
        self.Bind(wx.EVT_MENU, self.NewApplication, id=ID_NEW)
        self.Bind(wx.EVT_MENU, self.OpenFile, id=ID_OPEN)
        self.Bind(wx.EVT_MENU, self.SaveFile, id=ID_SAVE)
        self.Bind(wx.EVT_MENU, self.QuitApplication, id=ID_EXIT)
        self.Bind(wx.EVT_MENU, self.OnUndo, id=wx.ID_UNDO)
        self.Bind(wx.EVT_MENU, self.OnRedo, id=wx.ID_REDO)
        self.Bind(wx.EVT_MENU, self.OnCut, id=ID_CUT)
        self.Bind(wx.EVT_MENU, self.OnCopy, id=ID_COPY)
        self.Bind(wx.EVT_MENU, self.OnPaste, id=ID_PASTE)
        self.Bind(wx.EVT_MENU, self.OnAll, id=ID_ALL)
        self.Bind(wx.EVT_MENU, self.About, id=ID_ABOUT)

    def toolBar(self):
        self.toolbar = self.CreateToolBar()

        self.toolbar.AddSimpleTool(ID_NEW, \
                             wx.Bitmap("icons/filenew.png"), u"建立新檔...")
        self.toolbar.AddSimpleTool(ID_OPEN, \
                              wx.Bitmap("icons/folder.png"), u"開啟舊檔...")
        self.toolbar.AddSimpleTool(ID_SAVE, \
                            wx.Bitmap("icons/filesave.png"), u"儲存檔案...")
        self.toolbar.AddSeparator()

        self.toolbar.AddSimpleTool(wx.ID_UNDO, \
                       wx.Bitmap("icons/undo.png"), u"還原...", ""u"還原...")
```

```
        self.toolbar.AddSimpleTool(wx.ID_REDO, \
                      wx.Bitmap("icons/redo.png"), u"重作...", ""u"重作...")
        self.toolbar.AddSeparator()

        self.toolbar.AddSimpleTool(ID_CUT, \
                               wx.Bitmap("icons/editcut.png"), u"剪下...")
        self.toolbar.AddSimpleTool(ID_COPY, \
                              wx.Bitmap("icons/editcopy.png"), u"複製...")
        self.toolbar.AddSimpleTool(ID_PASTE, \
                             wx.Bitmap("icons/editpaste.png"), u"貼上...")
        self.toolbar.AddSeparator()

        self.toolbar.AddSimpleTool(ID_EXIT, wx.Bitmap("icons/exit.png"),\
                                                          u"關閉文字編輯器")
        self.toolbar.Realize()

        # 功能連結
        self.Bind(wx.EVT_TOOL, self.NewApplication, id=ID_NEW)
        self.Bind(wx.EVT_TOOL, self.OpenFile, id=ID_OPEN)
        self.Bind(wx.EVT_TOOL, self.SaveFile, id=ID_SAVE)
        self.Bind(wx.EVT_TOOL, self.OnUndo, id=wx.ID_UNDO)
        self.Bind(wx.EVT_TOOL, self.OnRedo, id=wx.ID_REDO)
        self.Bind(wx.EVT_TOOL, self.QuitApplication, id=ID_EXIT)
        self.Bind(wx.EVT_TOOL, self.OnCut, id=ID_CUT)
        self.Bind(wx.EVT_TOOL, self.OnCopy, id=ID_COPY)
        self.Bind(wx.EVT_TOOL, self.OnPaste, id=ID_PASTE)

    def textControl(self):
        self.text = wx.TextCtrl(self, ID_TEXT, style=wx.TE_MULTILINE)

        # 功能連結
        self.text.Bind(wx.EVT_TEXT, self.OnTextChanged, id=ID_TEXT)

    def statusBar(self):
        self.statusbar = self.CreateStatusBar()
        self.statusbar.SetFieldsCount(3)
        self.statusbar.SetStatusWidths([-5, -2, -1])

    def NewName(self):
        dialog = wx.TextEntryDialog(None, u"請輸入所要建立的文字檔名：", \
                                        u"建立新檔", "new.txt", style=wx.OK)
        if dialog.ShowModal() == wx.ID_OK:
            self.filename = dialog.GetValue()
        dialog.Destroy()

    def NewApplication(self, event):
        new = SEditor(None, -1, u"新的簡單文字編輯器")

    def OpenFile(self, event):
        self.dirname = os.getcwd()
        file_dialog = wx.FileDialog(self, u"選擇檔案...", self.dirname,\
                                                          style=wx.OPEN)
        if file_dialog.ShowModal() == wx.ID_OK:
            self.filename = file_dialog.GetFilename()
            self.dirname = file_dialog.GetDirectory()
            f = open(os.path.join(self.dirname, self.filename), "r")
            self.text.SetValue(f.read())
            f.close()
        file_dialog.Destroy()
```

```
    def SaveFile(self, event):
        if self.filename:
            f = open(self.filename, "w")
            text = self.text.GetValue()
            f.write(text)
            f.close()
        else:
            alert_dialog = wx.MessageDialog(None, \
                                u"因為發生不明原因，所以檔案不能儲存...", \
                                u"警告訊息", wx.OK | wx.ICON_ERROR)
            if alert_dialog.ShowModal() == wx.ID_OK:
                alert_dialog.Destroy()

    def QuitApplication(self, event):
        self.Close(True)

    def OnCut(self, event):
        self.text.Cut()

    def OnCopy(self, event):
        self.text.Copy()

    def OnPaste(self, event):
        self.text.Paste()

    def OnAll(self, event):
        self.text.SelectAll()

    def OnTextChanged(self, event):
        self.modify = True
        self.statusbar.SetStatusText(u"編輯本文...", 1)

        self.lines = self.text.GetLastPosition()
        self.statusbar.SetStatusText(unicode(self.lines), 2)

        event.Skip()

    def OnUndo(self, event):
        self.text.Undo()

    def OnRedo(self, event):
        self.text.Redo()

    def About(self, event):
        about_dialog = wx.MessageDialog(self, message, u"SEditor 版本資訊",\
                                                                    wx.OK)
        if about_dialog.ShowModal() == wx.ID_OK:
            about_dialog.Destroy()

if __name__ == "__main__":
    app = wx.App()
    frame = SEditor(None, -1, u"簡單的文字編輯器")
    app.MainLoop()
```