# **创建一个wx.App子类**

**需要4步来完成：**

1. **定义app子类和frame子类**
2. **在app子类中写一个OnInit()方法**
3. **在程序的主要部分创建这个类的实例**
4. **调用程序的MainLoop()方法，这会把程序的控制权交给wxpython**

**Note:**

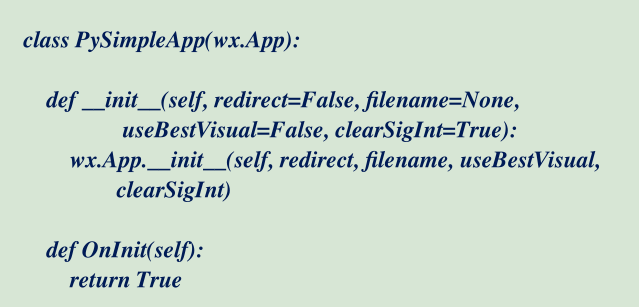
**OnInit()方法不要求参数且返回一个布尔值，如果返回的是Fasle，程序立即退出。大多数情况下需要返回真，但当需要处理错误条件时，退出可能是更好的方法**

**由于OnInit()方法是wxpython架构的一部分，所以定制类的初始化都由它来完成，而不是\_\_init\_\_方法，如果想要使用\_\_init\_\_,一定要记住继承wx.App.\_\_init\_\_()。**

**通常，在OnInit()方法中要至少创建一个框架对象，并调用show方法，也可以有选择的调用settopwindow方法来指定一个框架作为顶级窗口，它将作为那些没有指定父框架的对话框的默认父窗口**

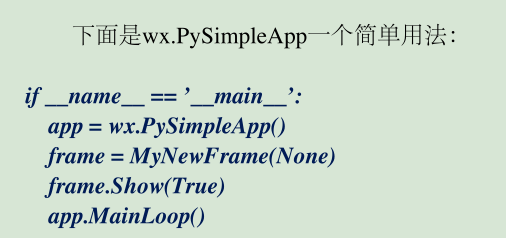
## **懒惰一下：不用创建wx.app的子类？**

**如果只有一个框架，就不需要自己创建一个子类，因为wx提供了一个这样的类,wx.PySimpleApp,下面是它的源代码：**

****

**看到了吗，它只是返回true，什么都没做**

**下面是一个用法：**

****

# **生命周期**

**在wx中，程序的生命周期开始于应用程序对象被创建时，结束在应用程序对象被注销后。在这个过程中发生什么呢？**

**就像之前的程序一样，框架对象的创建是放在应用程序对象的OnInit中的，因为要先创建应用对象，然后再创建框架对象，之后进入MainLoop循环，处理相应的事件，当所有的窗口被关闭，则应用程序对象被注销，生命周期结束，最后程序会关闭其他的连接的线程。**

**框架对象不能在应用程序对象被创建之前创建，因为这样的话就跳出了生命周期，框架就“死了”。**

## **重定向输出**

**如果wx控制了标准流，那么经由任何方法发送到流的文本都将会被发送到一个wx框架上，在生命周期之外的标准流都会交给python控制台处理，而生命周期之内的，都会被重定向到一个框架里面，程序02.py演示了这个功能。**

## **关闭wx应用程序**

**正常的关闭：可以在app类中定义一个OnExit()方法，在所有窗口关闭后且在wx内在的清理程序之前被调用，可以用这个方法来清理任何你创建的非wx资源，比如一个数据库连接等，即使使用wx.Exit()方法，OnExit()方法依然会被调用。当窗口关闭，控制权交还给python控制台时，onexit（）方法就会被调用，用来处理python的一些其他程序所创建的线程。**

**但是如果想要在最后的窗口被关闭后程序依然可以运行，那需要wx.App的SetExitOnDelete(flag)方法来改变默认的行为。如果为True,则最后的窗口被关闭后程序依然运行，这意味着wx.app实例依然“活着”，wx应用程序一直存活到wx.Exit()被调用**

**立即关闭和紧急关闭：在有些时候，你需要立即结束程序而不需要清理工作，这里有两种方法来在紧急情况下退出你的wx程序。一是调用wx.App的ExitMainLoop()方法。这个方法会显式的使用主消息循环终止，使用控制离开MainLoop()函数；二是调用全局的wx.Exit()方法。这两种方法都不推荐，因为它将导致一些清理函数被跳过。**

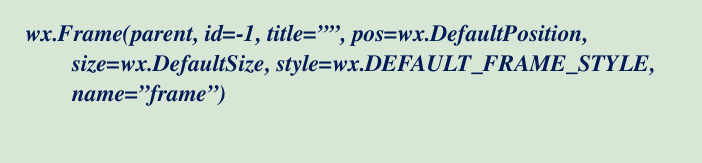
## **顶级窗口**

**一般意义上的顶级窗口指的是没有任何父容器的窗口，一个应用程序至少有一个顶级窗口，可以有多个，但必须指定一个主顶级窗口，使用settopwindow()方法，如果不指定顶级窗口，那么在wx.app中的顶级窗口列表中的第一个框架会被认为是主顶级窗口。没有必要明确的定义一个主顶级窗口，特别是只有一个窗口时。**

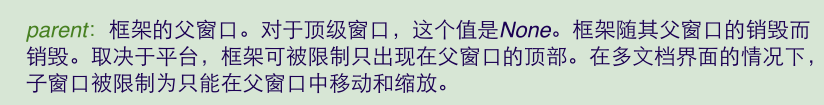
## **Wx.Frame()类**

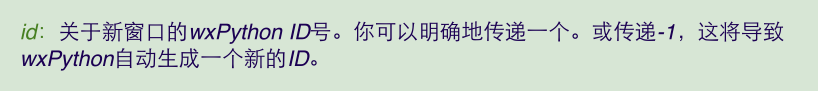
**Wx.Frame是所有框架的父类，我们可以自己创建框架，但这里也有一些预备的frame模板，作为wx.Frame的子类；**

**Wx.Frame.init()的构造器方法是这样的**

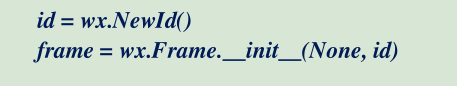
****

**参数解释：**

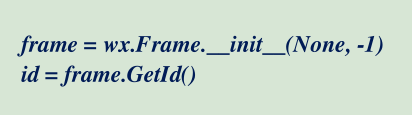
****

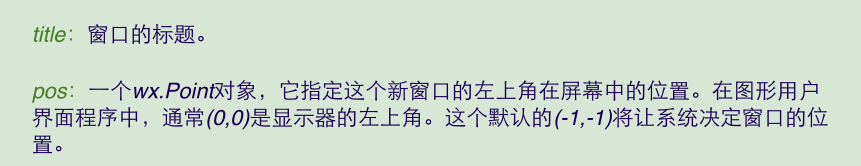
****

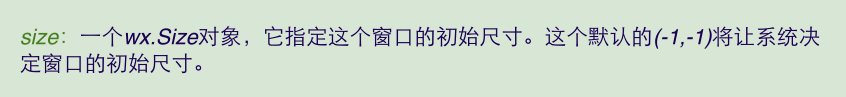
**自己创建id号非常麻烦，有可能重复，使用wx.newid来让程序自动创建id**



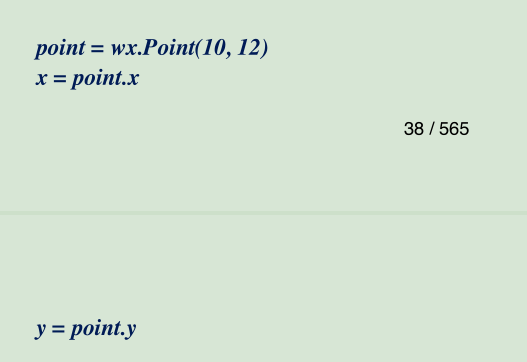
当你需要这个id时，可以使用getid来获得



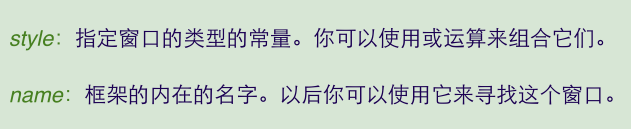
****

****

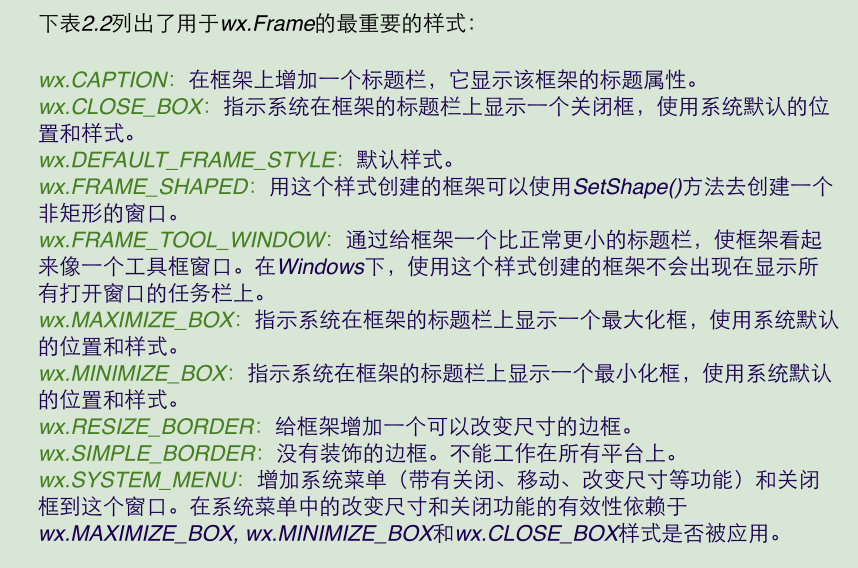
**我们可以使用get和set函数来获取位置参数，getsize返回一个元祖，可以像下面这样访问。Point类表示一个点或者位置，默认是0**

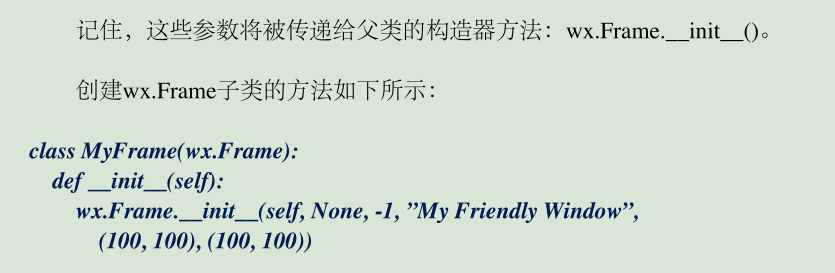


**Size类和point类一样，都可以这样用，这些都会在后面学习到**

****

**样式有以下几种**



****

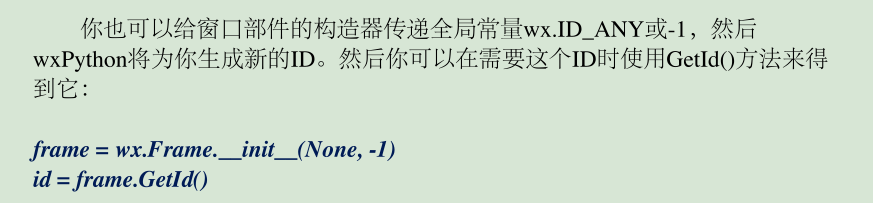
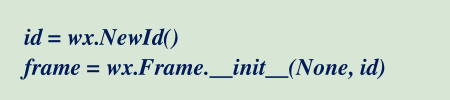
## 使用wxpython的ID

**在wx中，ID是所有窗口的标识，每个窗口都有唯一的标识，尽管可以重用id，但尽量保证在整个应用程序中id的唯一性；有一些ID是预定义的，比如wx.ID\_OK标识对话框中的ok。**

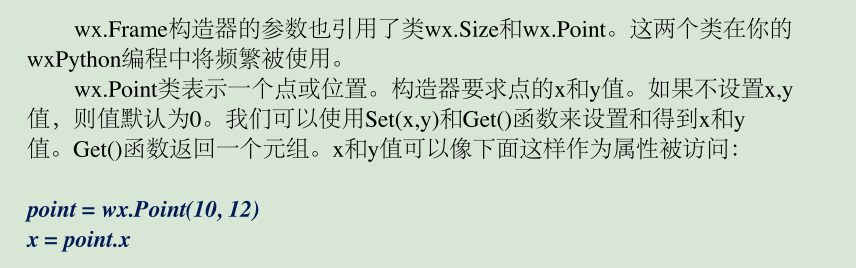
**ID号的作用：在指定的对象发生的事件和响应该事件的回调函数之间建立唯一的联系。**

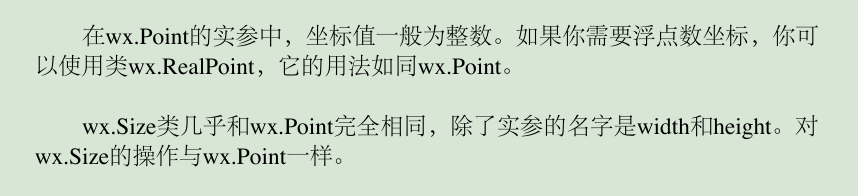
**创建ID的方法：**

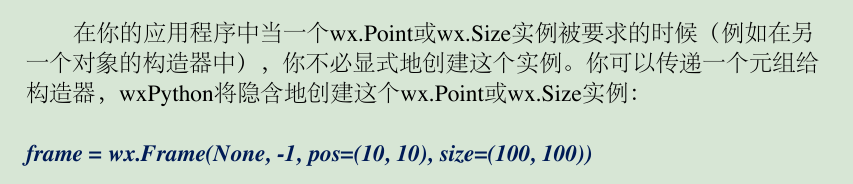
1. **指定正整数，但是不能是4999-5999之间的数，也不能是已经被重用了的常量。可以调用wx.RegisterID()来确保wx不会再别处使用你的ID**
2. **Wx.NewId(),自己确保ID的唯一性非常麻烦，可以使用函数来完成；**

****

## **使用wx.Size和wx.Point**

****

****

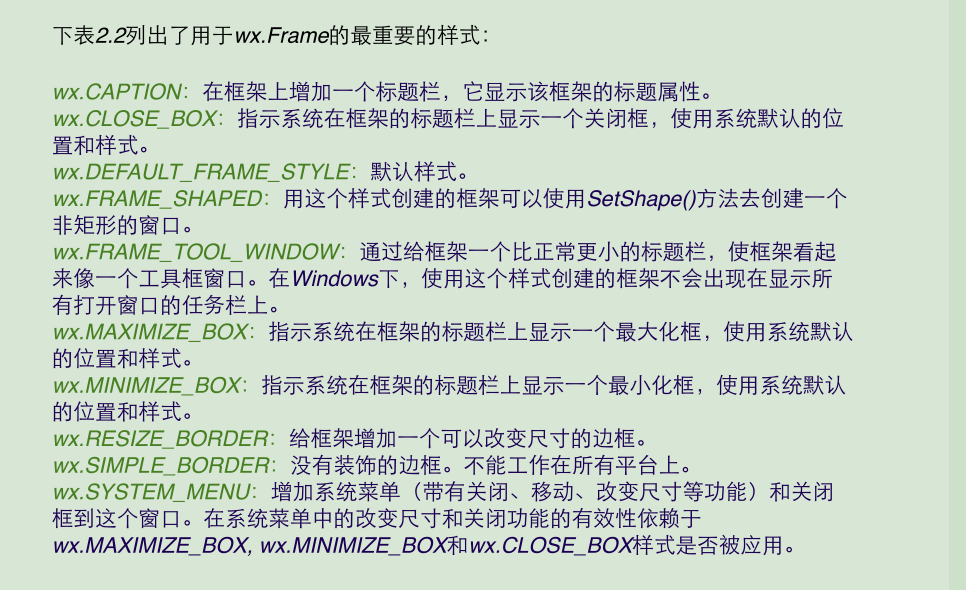
****

## **使用wx.Frame样式**

**每个wx窗口都会有一个样式参数，wx提供了一些重要的样式，而这些样式其实都是多种样式的集合**

****

**一些常用样式**

****

## **为框架创建一个对象和子窗口**

**之前讲了如何创建wx.frame对象，但这只是空的框架，这里讨论如何插入对象和子窗口。**

**02.py的第二个程序示范了如何创建一个对象**

****

**在这个程序实例中，使用了一个panel对象作为框架的子窗口对象，没有必要为所有的子窗口对象显示的指定位置和大小，这没什么意义，因为一旦调整框架的大小，相应的位置和大小却没有变化，这会让界面更难看。所以wx会自动调整它，不用再指定这些东西。Wx使用叫sizers的对象来管理子窗口的布局；**

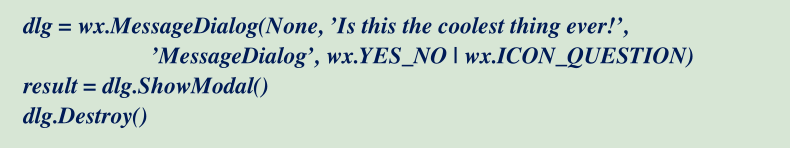
**不需要显示的插入父窗口来明确窗口之间的关系，只需在子窗口被创建时指定父窗口就可以了；**

**02.py中的第三个程序展示了如何给窗口增加一个菜单栏，状态栏**

****

## **消息对话框**

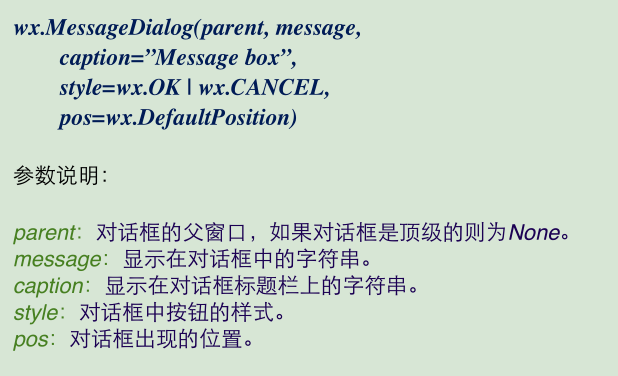
**与用户通信的最基本机制是wx.MessageDialog，它是一个简单提示对话框，可以用作简单的yes/no的对话，代码如下图：**

****

**02.py中的第四个程序显示了如何使用消息对话框**

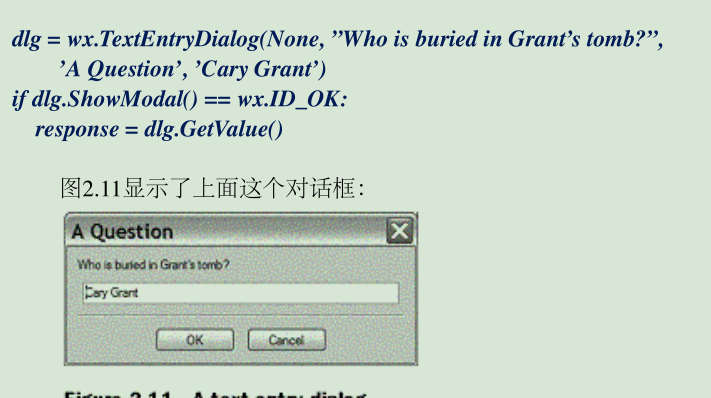
**Showmodal()会让对象以模式框架的形式显示，这意味着用户在对话框关闭之前，不能响应其他的事件。这个函数的返回值是一个整数，对于messagedialog类来说，是wx.ID\_YES，wx.ID\_NO，wx.ID\_CANCEL，wx.ID\_OK之一。**

**Wx.MessageDialog的参数如下：**

****

## **文本输入框**

**下面的代码显示如何显示文本输入框，当用户按下yes时，会捕捉到用户的输入**

****

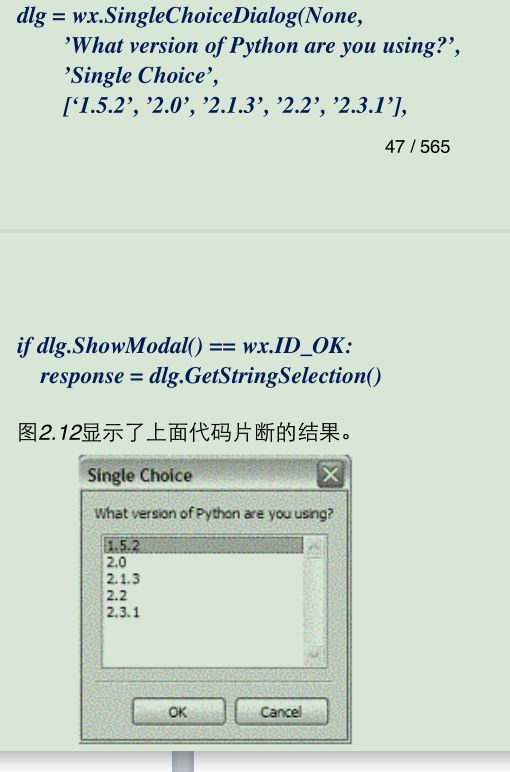
**列表中的参数依次说明**

**父窗口，显示的内容，标题(默认是please enter text:)，文本框中的默认值；**

**同上面一样showmodal()同样得到一个整数，是四个之一；**

**Getvalue()得到用户的输入，同样，setvalue()可以改变用户输入**

## **从列表中选择**

****

**wx.SingleChoiceDialog的参数和文本输入框的基本一样，除了用列表代替了文本字符串。**

**得到所选的结果有两种方法：getstringselection()返回实际选择到的结果；getselection()得到用户选项的索引**