

Glade

A Tutorial by
Paul Hogan October 2006

p.j.hogan@open.ac.uk - pachjo@hotmail.co.uk



Table of Contents

i	Introduction	1
ii	A Note on Terminology	2
1	Glade The Starting Point	3
2	Gedit The Starting Point	9
3	Python Run #1	11
4	Glade Configure Window1	13
5	Gedit Connect The Window1 Destroy Signal	17
6	Python Run #2	18
7	Glade Add A Button	19
8	Python Run #3	22
9	Glade Configure The Button	23
10	Python Run #4	27
11	Glade Complete The GUI	28
12	Python Run #5	30
13	Gedit Complete The Python Code	31
14	Python Run #6 The Finished Application	32
15	Final Words	34

Introduction

I have written this tutorial to show how I learned to grasp how one can create a GUI application under Linux using Glade, Gedit and Python.

Coming from Windows I found the way GUIs are created under Linux totally alien, time consuming and very frustrating especially being used to development environments such as MS Access etc.

The terminology was also confusing, but this particular part of the transition was very quickly overcome.

There is a wealth of help out there to teach you how to install applications required so I won't cover this here. I will however let you know that in order to run through this tutorial you need to have the following packages installed:

Python 2.4 or less and associated libraries
Glade or Glade Gnome and associated libraries
Gedit

To make life easier I have my environment set up so each development tool is on a different desktop. This makes it easy to switch between each running development tool as we create our application. Therefore open now Glade, Gedit, Python and a console each on a different desktop.

The approach used in this tutorial is to cycle through each stage of development from GUI design to editing the python code to running the application.

So we go from Glade to Gedit to console and back to Glade and continue to cycle this way until we have complete our application.

A Note on Terminology

A GUI in Linux has two main types of objects, widgets and containers.

Widgets are windows, buttons, labels, combo boxes etc.
Containers are what holds the widgets.

A GUI in Linux has signals and callbacks (callback functions).

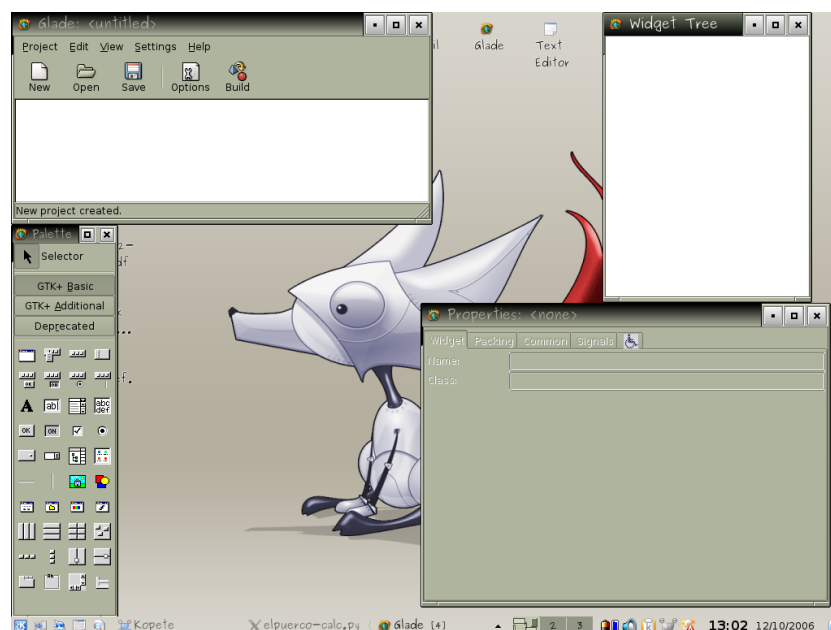
Signals are what are generated in response to an event, such as clicking on a button. This would generate an 'on_clicked' signal.

Callbacks are functions that respond to the signals.

Under Windows these are known as events and event handlers.

Glade The Starting Point

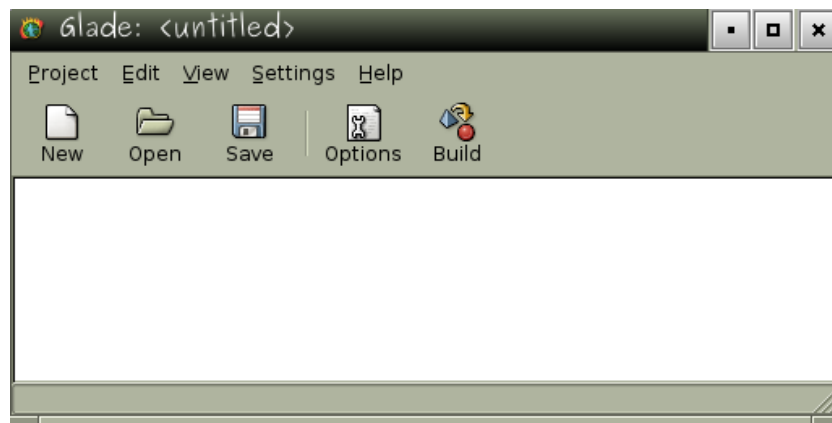
- 1.1 select the Glade desktop.
- 1.2 You should see the forms as detailed below, if you don't have the same layout then click the View menu on the window titled 'Glade: <untitled>' and tick every option except 'Show Clipboard'.
- 1.3 It is probably good practice to leave all four windows open so that is what we will do that.



1.4 The Main window

1.4.1 This window is the Main window which allows you to amongst other things open, save and set options for your projects.

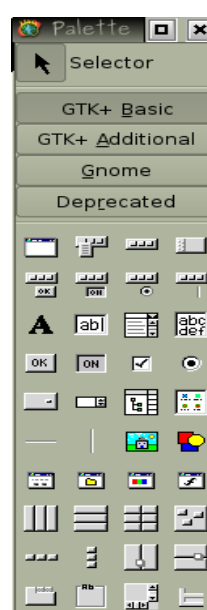
It also lists in the area under the toolbar all widgets in your application.



1.5 The Palette

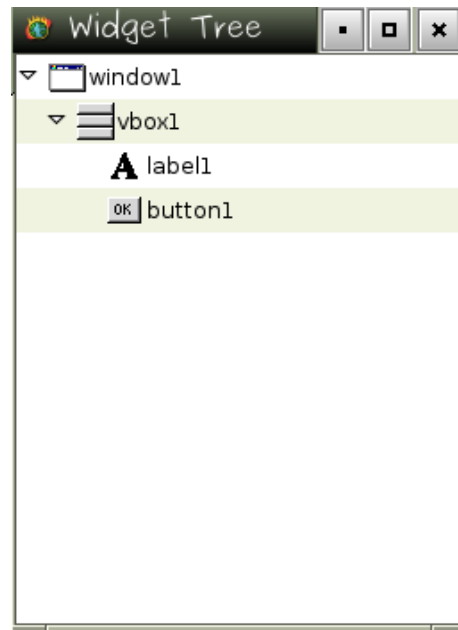
1.5.1 This allows you to select the widgets and containers to use in your application.

The palette on the right has a Gnome button, this will only be visible if you have installed glade gnome.



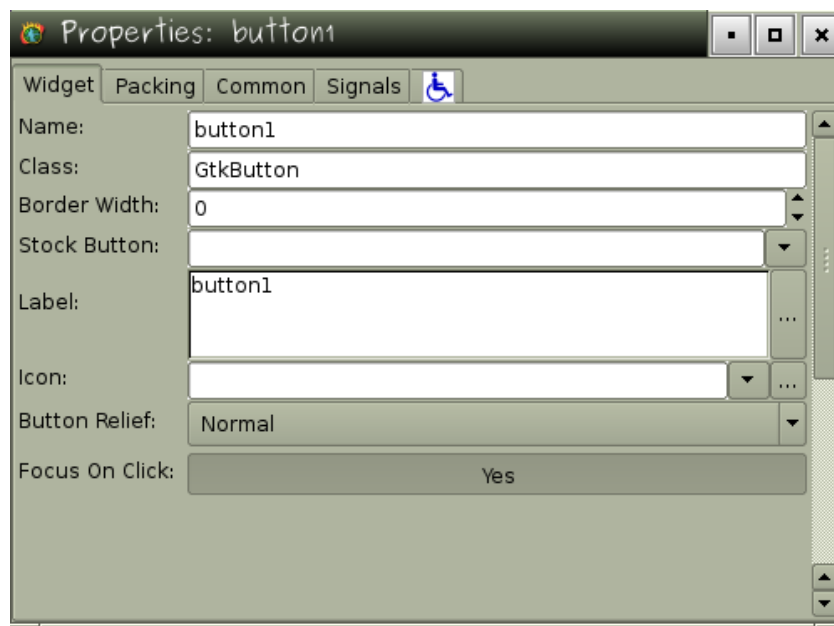
1.6 The Widget Tree

1.6.1 This window displays a hierarchical view of all widgets and containers in your application.

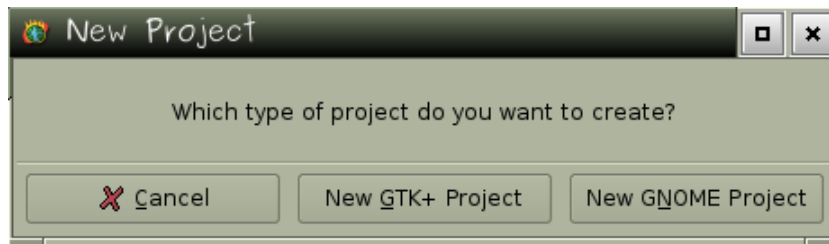


1.7 The Properties Window

1.7.1 This window allows you to configure various settings of your widgets and containers.



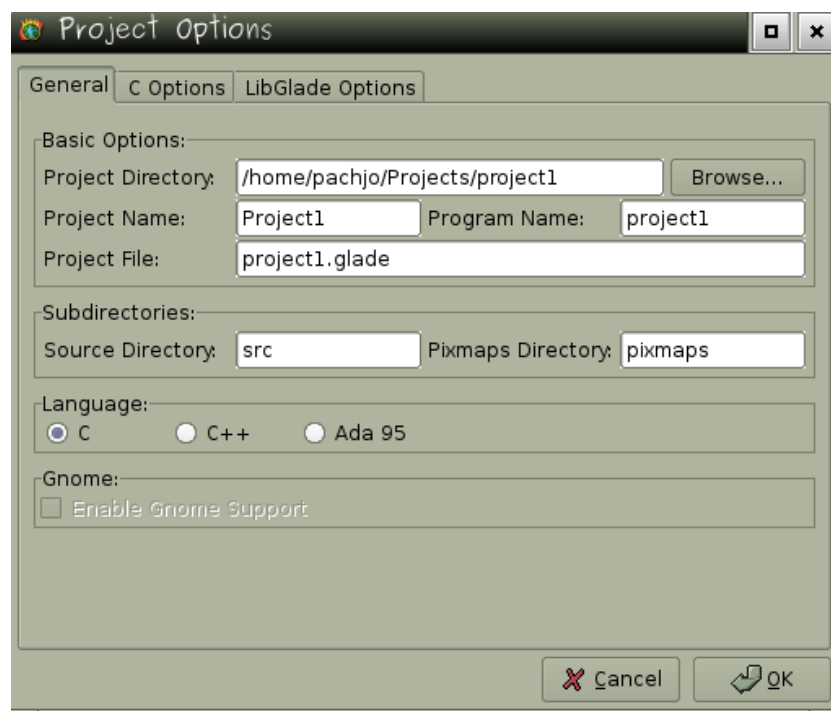
- 1.8 Click New on the Main window to display the New Project dialog.



- 1.9 Click on the New GTK+ Project button.

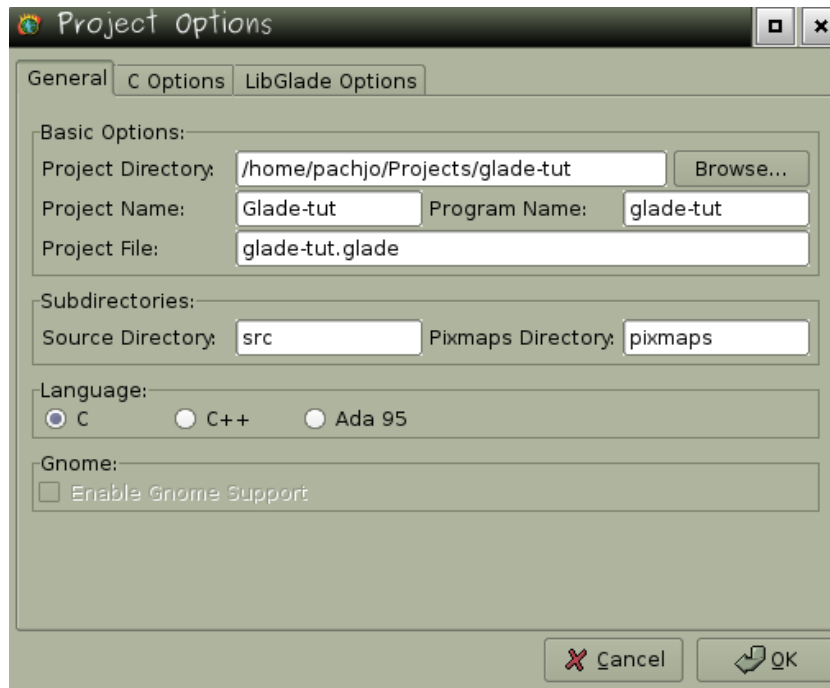
- 1.10 Click the Save button on the Main window toolbar to display the Project Options dialog.

By doing this we setup the required directory to house all the files for our application.

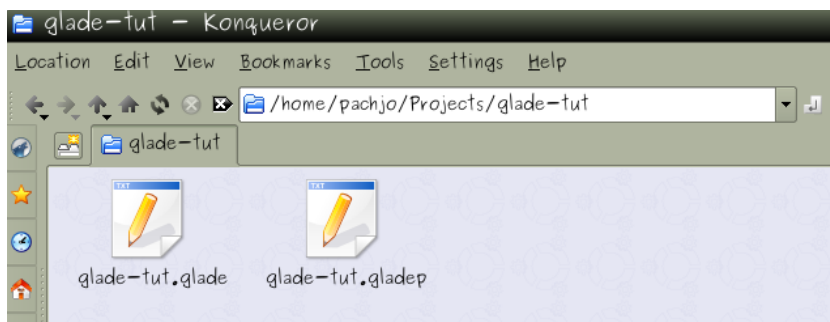


- 1.11 Change the last entry in the Project Directory field from project1 to glade-tut.

- 1.12 Notice that Project Name and Program Name change to glade-tut too.



- 1.13 Click the OK button to save the project.
- 1.14 View the contents for the project directory you just created. Two files have been created, glade-tut.glade and glade-tut.gladep



The .glade file is the XML file that we will use with Python, don't be concerned with this file as everything we do with it will be done with Glade.

The .gladep file is the Glade project file, again everything we do with it will be done through Glade.

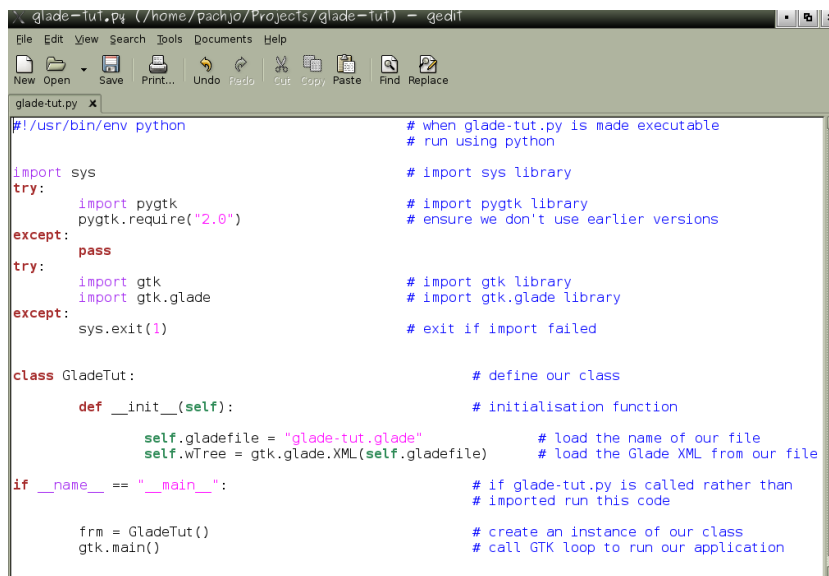
- 1.15 Go back to the Glade desktop and click the GTK+ Basic button on the Palette.
- 1.16 Click the Window button on the Palette. If you hover your mouse over the buttons you will see a tool tip which will aid your locating the correct button.
- 1.17 A new window will be displayed with the title 'window1'.



- 1.17.1 The Main window will display window1 in its display area
 - 1.17.2 The Widget Tree will display window1
 - 1.17.3 The Properties window will display the properties for window1
- 1.18 Click the Save button on the Main window toolbar to save the changes.
 - 1.19 This is all we need to do for now as far as creating a basic no thrills GUI that does absolutely nothing goes. The next section will detail how to create the Python file we need to run our application.

Gedit The Starting Point

- 2.1 select the Gedit desktop and select the Gedit window.
- 2.2 Before doing anything save the file as `glade-tut.py` in the same directory that contains our Glade files. Doing this means that Gedit will now help us by using Python syntax highlighting as we type our code.
- 2.3 Enter the following code into a new document.



```
#!/usr/bin/env python                                # when glade-tut.py is made executable
                                                    # run using python

import sys                                           # import sys library
try:
    import pygtk
    pygtk.require("2.0")
except:
    pass
try:
    import gtk
    import gtk.glade
except:
    sys.exit(1)                                     # exit if import failed

class GladeTut:                                     # define our class
    def __init__(self):                             # initialisation function
        self.gladefile = "glade-tut.glade"          # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile)   # load the Glade XML from our file

if __name__ == "__main__":                          # if glade-tut.py is called rather than
                                                    # imported run this code
    frm = GladeTut()
    gtk.main()                                     # create an instance of our class
                                                    # call GTK loop to run our application
```

- 2.4 This is not a Python guide so I will only explain the bits to do with Glade, there are plenty of Python tutorials out there should you need further reading.

2.4.1 The import statements import required libraries.

2.4.2 GladeTut is our class which at the moment contains only the `__init__` function. This function gets called when an instance of this class is created. Our Glade file is loaded which defines our GUI.

2.4.3 The if block checks to see if this code is being called by Python or as part of an import statement. If Python is calling it then an instance of the GladeTut class is created and the the `GTK.Main()` function is called to run our application.

2.5 Click the Save button.

2.6 Now we can run it and see what our application looks like, the next section will detail how we run our application to see what it looks like so far.

Python Run #1

3.1 Select the console desktop and click in the console window.

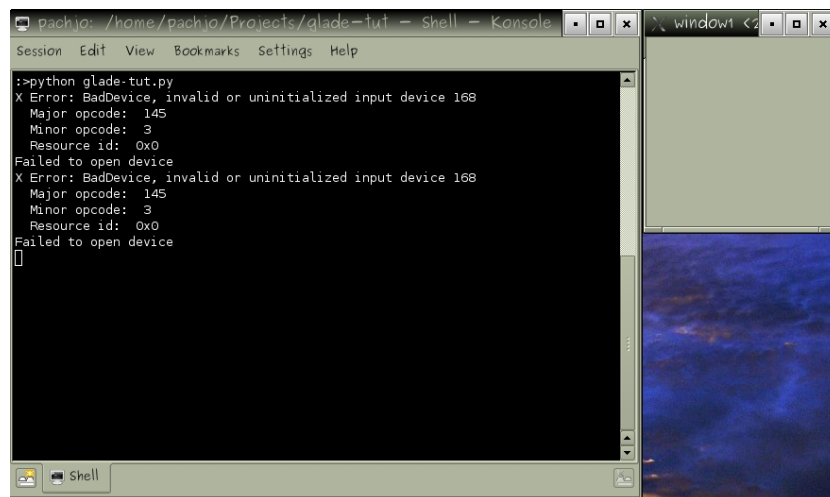
3.2 Change to the project file directory, this is my command.

```
cd /home/pachjo/Projects/glade-tut
```

3.3 Type the following command to run our application.

```
python glade-tut.py
```

3.4 You should see a small window appear as detailed below, however your window may appear in a different place on your desktop.



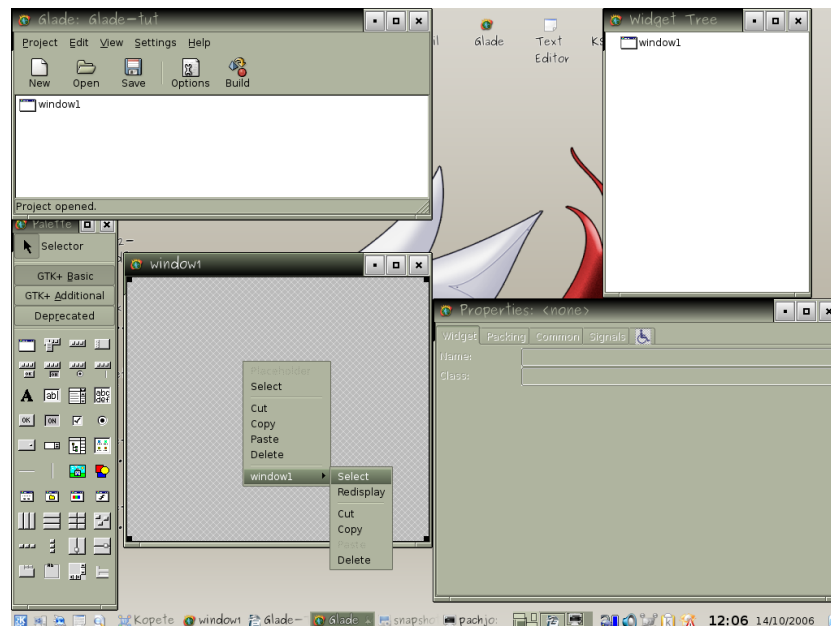
3.5 So now we have a running application that does nothing at all, but if you play around with the window you will see it has full GUI functionality. You can minimize and resize it just like any other GUI.

3.6 Close the window by clicking on the X on the top right corner.

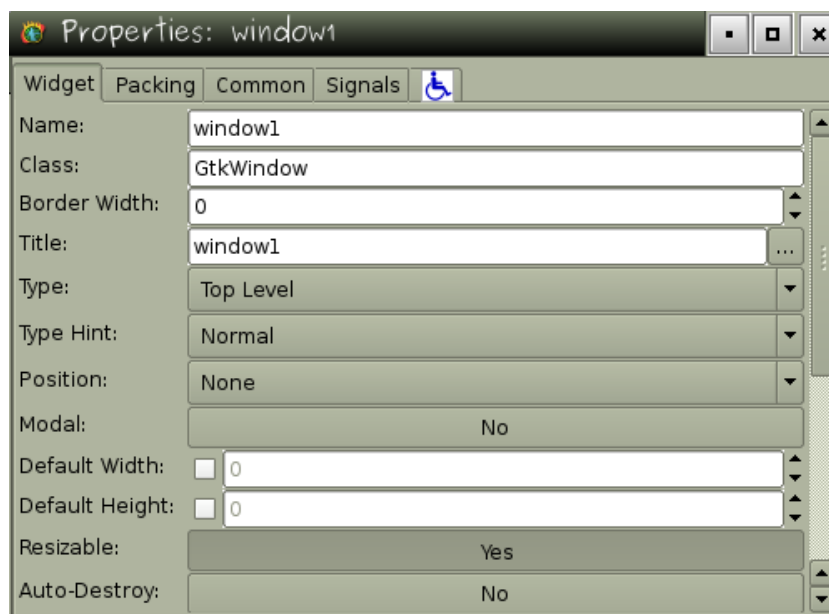
- 3.7 And here we see our first problem, the window has closed but we don't get our console prompt back! This is because we have not told GTK to stop running so to get our prompt back press `ctrl-c`
- 3.8 So now we have our prompt back we have completed the first cycle of our development as detailed in the introduction, namely, from Glade to Gedit to console. And now we go back to Glade to sort out getting our application to close neatly when we click the X button.

Glade Configure Window1

- 4.1 select the Glade desktop.
- 4.2 Ensure that the Properties window displays the properties for window1. Right click anywhere in window1 and from the popup menu select the option window1 and then the option select.



- 4.3 The Properties window will now display the properties of window1



4.4 To save a little time we will modify a few settings in one go and I will explain what they are and what they do. First off click on the Widget tab.

4.4.1 Change Title: to Glade-Tut, this will change the window title from window1 to Glade-Tut.

4.4.2 Change Position: to Centre, this will make our window appear in the centre of our desktop.

4.4.3 Change Resizeable: to No.

4.5 Click on the Common tab.

4.5.1 Click the Width checkbox to place a tick in it and then enter the value 300. This will set the width of our window.

4.5.2 Click the Height: checkbox to place a tick in it and then enter the value 200. This will set the height of our window.

4.6 Click the Signals tab.

4.6.1 Click the button with the label ... to open the Select signal dialog.

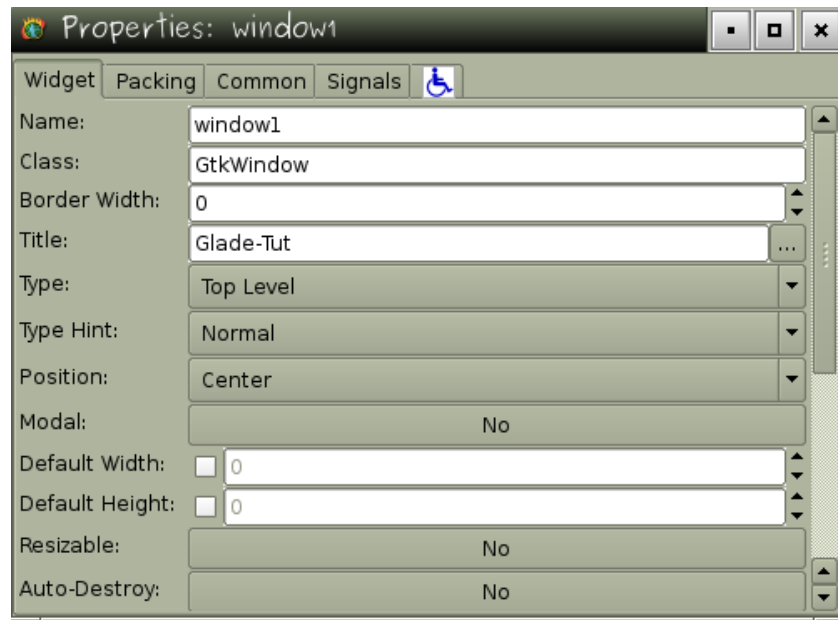
4.6.2 Scroll all the way to the bottom until you see the heading GTKObject signals and select the option destroy then click the OK button.

4.6.3 Back at the Signals tab click the Add button. An entry will appear in the display area of the Signals tab. The signal column will show destroy and the Handler column on_window1_destroy.

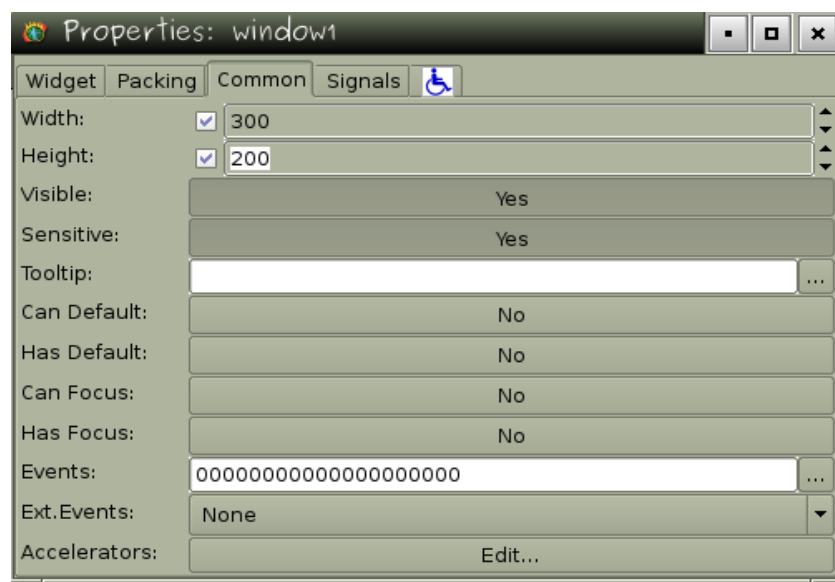
4.7 Click on the Save button on the Glade Main window.

4.8 Your window1 Properties window should now look like this.

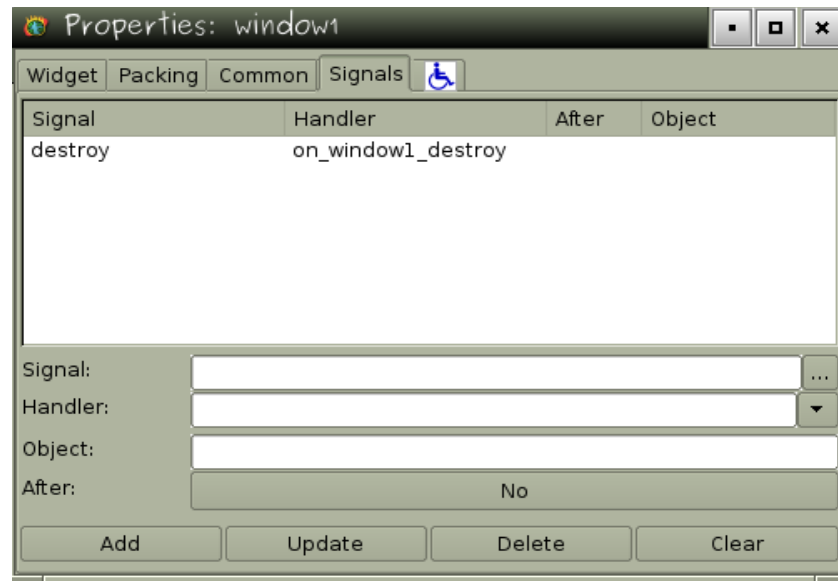
4.8.1 The Widget tab



4.8.2 The Common tab



4.8.3 The Signals tab



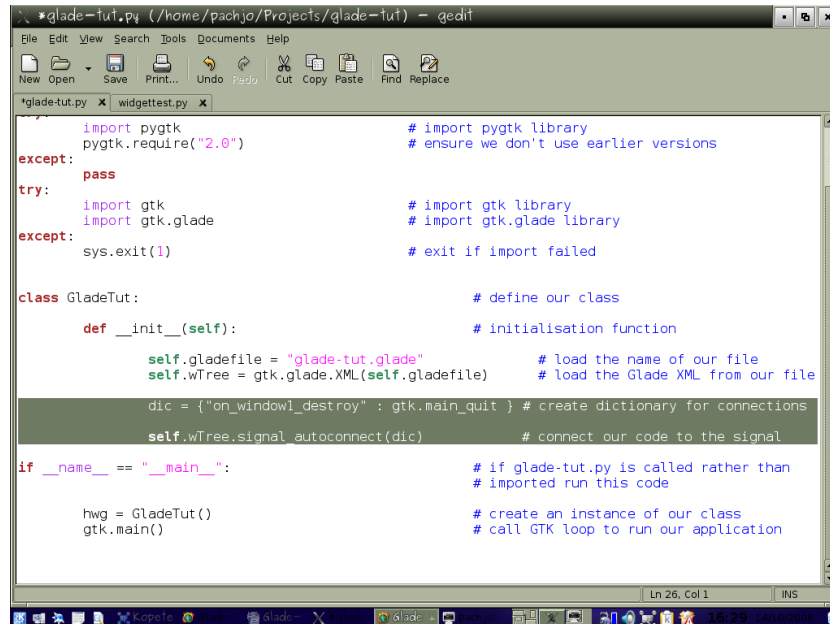
4.9 The Signals tab requires a little explaining. Back in the section titled 'A Note on Terminology' you read about signals and callbacks. The Signals tab is where you define the signals that you want your Python code to respond to.

We have setup a single signal here to respond to the `window1` `destroy` event. When the window is closed the `destroy` signal is emitted and the callback function that responds to that signal is called `on_window1_destroy`.

4.10 We shall now code in our `glade-tut.py` file the callback function `on_window1_destroy`.

Gedit Connect The Window Destroy Signal

- 5.1 Select the Gedit desktop and select the Gedit window.
- 5.2 Amend the `--init__(self)` function to look like this.



```
*glade-tut.py (/home/pachjo/Projects/glade-tut) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace

*glade-tut.py x widgettest.py x

import pygtk                                # import pygtk library
pygtk.require("2.0")                        # ensure we don't use earlier versions
except:
    pass
try:
    import gtk                                # import gtk library
    import gtk.glade                          # import gtk.glade library
except:
    sys.exit(1)                              # exit if import failed

class GladeTut:                              # define our class
    def __init__(self):                      # initialisation function
        self.gladefile = "glade-tut.glade"  # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile) # load the Glade XML from our file

        dic = {"on_window1_destroy": gtk.main_quit} # create dictionary for connections
        self.wTree.signal_autoconnect(dic)    # connect our code to the signal

if __name__ == "__main__":                  # if glade-tut.py is called rather than
    # imported run this code
    hmg = GladeTut()                        # create an instance of our class
    gtk.main()                              # call GTK loop to run our application

Ln 26, Col 1 INS
```

- 5.3 The first highlighted line creates a dictionary to hold the details of the connection we want to make and second calls the `signal_autoconnect()` function to make the connection.

When the window is closed the destroy signal fires and the `on_window1_destroy` callback gets called which as you can see will execute the `gtk.main_quit()` function to close our application.

- 5.4 That is all that is required to connect our code to the signals of our window1 in the GUI.
- 5.5 Click on the Save button.
- 5.6 Now it is time to test what the changes look like.

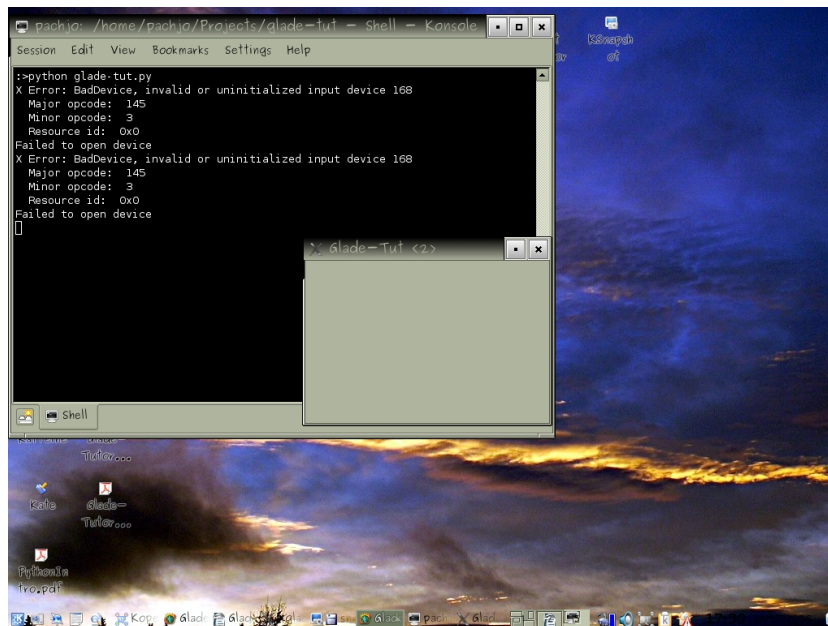
Python Run #2

6.1 select the console desktop and click in the console window.

6.2 Type the following command to run our application.

```
python glade-tut.py
```

6.4 Our window should now appear in the centre of the desktop.



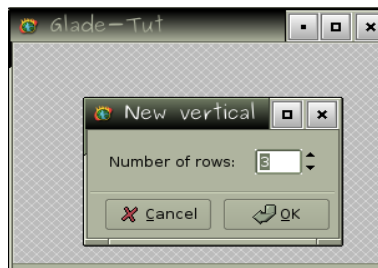
6.5 You will see the window title is now Glade-Tut, that the window is the size we set and that it is now not resizeable.

6.6 Close the window and see that the prompt returns as we setup the destroy signal to ensure our application closes correctly.

6.7 Now lets get to adding some bells and whistles to our application.

Glade Add A Button

- 7.1 Select the Glade desktop.
- 7.2 Ensure the GTK+ Basic button is selected on the Palette.
- 7.3 Click on the Vertical Box button then click anywhere on the window1 window to display the row setup dialog.

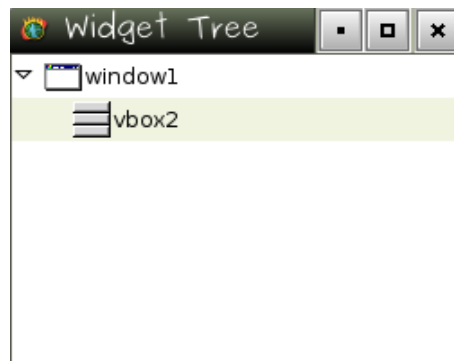


- 7.4 Leave the default setting of 3 then click the OK button.
- 7.5 A vertical container will appear on window1 with three rows.



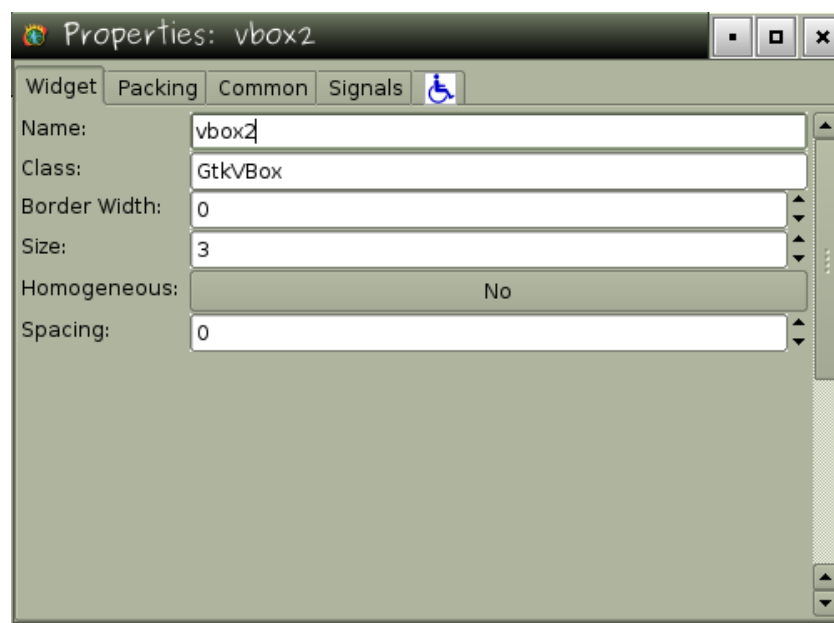
- 7.6 Each row of the vertical box, which is a container, can be selected individually. Try clicking in each row to see this in action.

- 7.7 Click on the little triangle to the left of the window1 entry in the Widget Tree window to expand the entry.



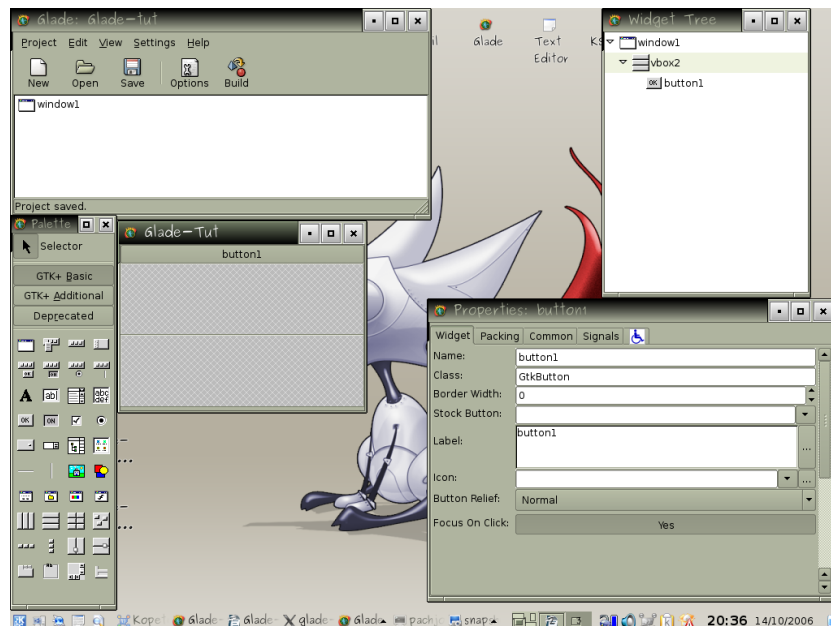
- 7.8 Click on the vbox entry and see that its properties are then displayed in the Properties window.

Your display will may display a different name for the vbox.



- 7.9 For the time being we won't change any of the properties as it will make more sense once we place a widget in it.
- 7.10 Ensure the GTK+ Basic button is selected on the Palette.

- 7.11 Click on the Button button then click anywhere in the top row of the vbox in window1.
- 7.12 Your Glade desktop should now look like this.



- 7.12.1 window1 has a new button in the top row of the vbox.
- 7.12.2 The Widget Tree window shows the new button in the listing.
- 7.12.3 The Properties window shows the properties for the new button.
- 7.13 Click on the Save button of the Glade Main window.
- 7.14 Before we start to change the properties for the vbox container and button widget lets run the application again to see how it looks now.

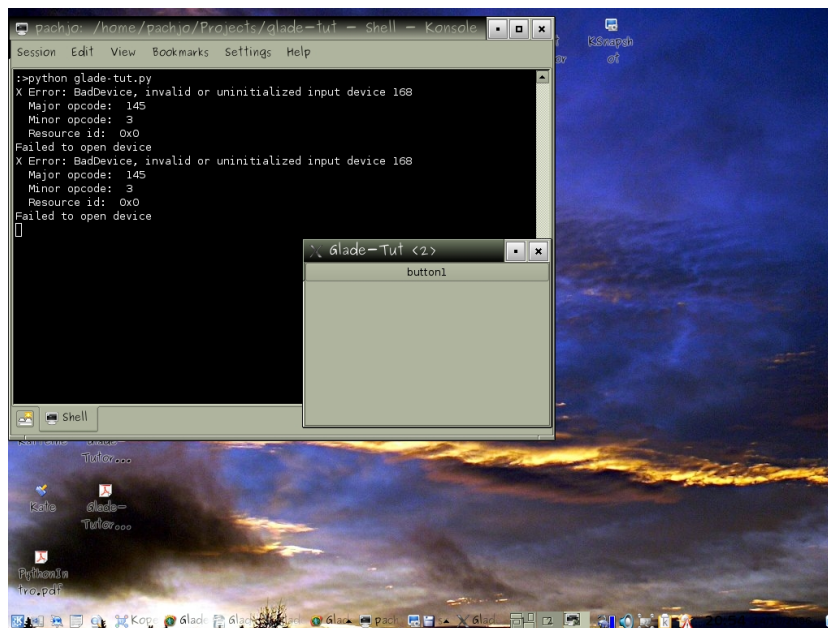
Python Run #3

8.1 select the console desktop and click in the console window.

8.2 Type the following command to run our application.

```
python glade-tut.py
```

8.3 Our window should now look like this.



8.4 The button we added appears squashed at the top and stretched across the entire width of the window.

8.5 Like when we first created and ran our application for the first time, the button functions as expected in that it will depress when clicked. However as we have not yet setup a signal for the button nothing happens.

8.6 Lets now return to Glade to see how we can change the properties of the vbox and the button and add a signal to our button.

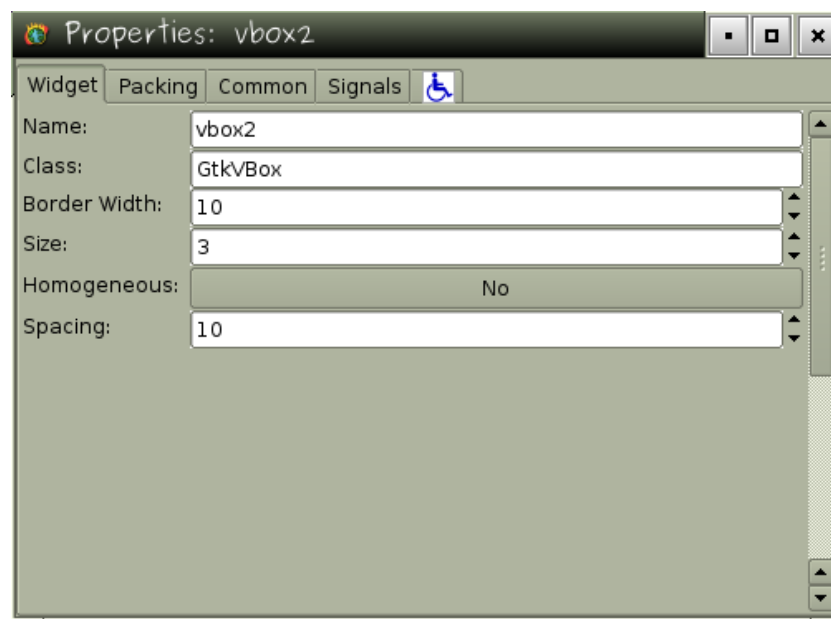
Glade Configure The Button

- 9.1 Select the Glade desktop.
- 9.2 Click on the button we placed in window1 so that its properties are displayed in the Properties window.
- 9.3 Change Name: to btnNumberOne.
- 9.4 Change Label: to Number One.
- 9.5 Click on the Packing tab.
- 9.6 Position: is currently set to 0 which means it is place in the first row of our vbox containers.
 - 9.6.1 Using the up and down arrows of the Position: property change the value from 0 to 1 and then to 2 and all the way back to 0 again and watch how the button changes position on our form.
- 9.7 Change Expand: to Yes.
- 9.8 Change Fill: to Yes.
- 9.9 btnNumberOne now fills the entire first row of the vbox.
- 9.10 Click on the Signals tab.
- 9.11 Click on the button labelled ... of the signal property.
- 9.12 select Clicked under the heading GTKButton signals then click the OK button.
- 9.13 Click the Add button on the signals tab to add the signal to the list.

- 9.14 Click on the vbox entry in the Widget Tree window so that its properties are displayed in the Properties window.
- 9.15 Click on the Widget tab.
- 9.16 Change Border Width: to 10.
- 9.17 Change Spacing: to 10.
- 9.18 window1 should now look like this.

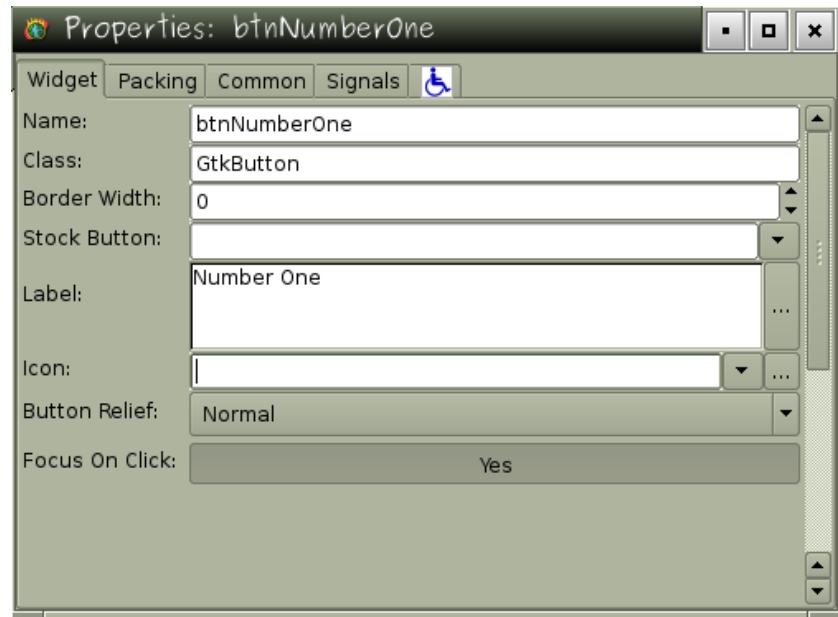


- 9.19 Now the button appears nicely spaced in our window.
- 9.20 Your vbox Properties window should look like this

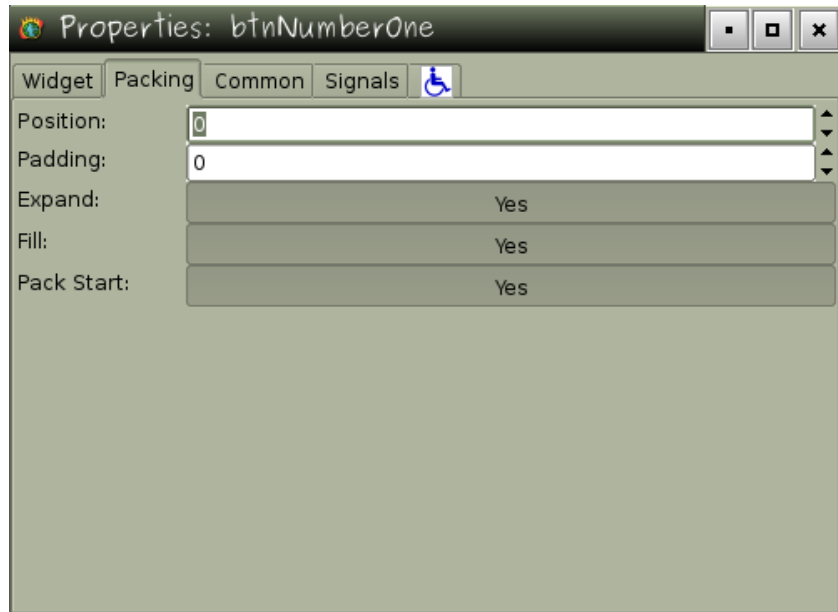


9.21 Your button Properties window should look like this.

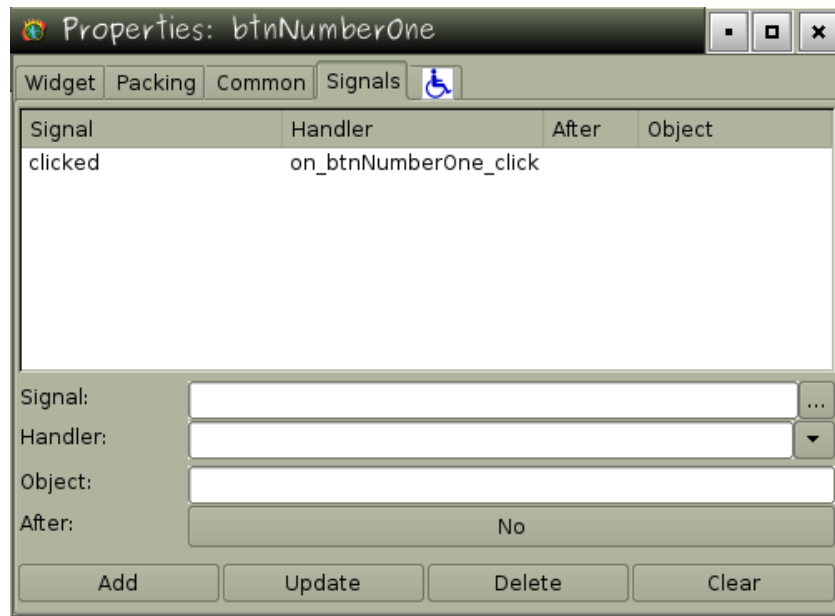
9.21.1 The Widget tab.



9.21.2 The Packing tab.



9.21.3 The signals tab.



9.22 Click the save button on the Glade Main window.

9.23 Now let's see what our application runs like.

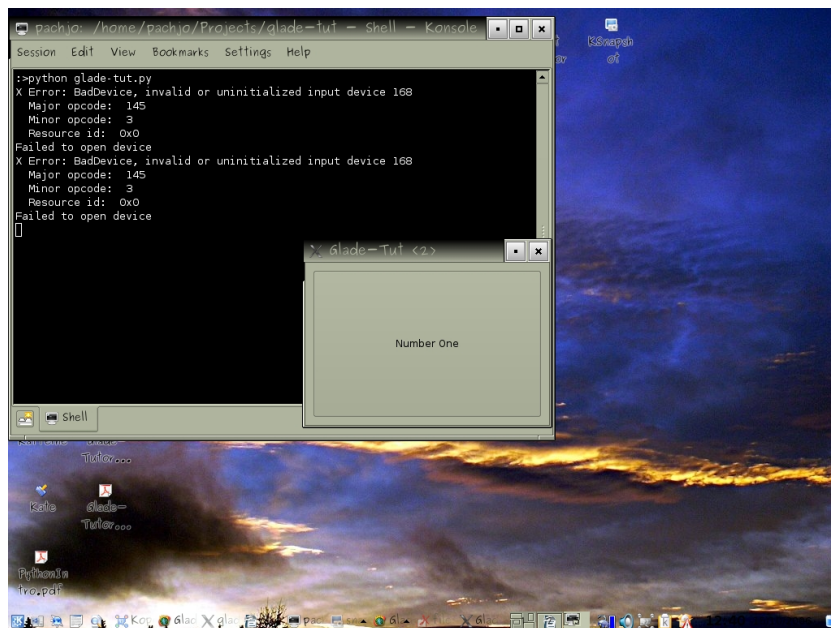
Python Run #4

10.1 select the console desktop and click in the console window.

10.2 Type the following command to run our application.

```
python glade-tut.py
```

10.3 Our window should now look like this.



10.4 Not exactly what we wanted, but we can see that our single button now has a neat border around it.

10.5 Lets get back to Glade and finish the GUI work.

Glade Complete The GUI

- 11.1 We are now going to complete our GUI setup as we now have the basics under our belt to help us to accomplish this.
- 11.2 Select the Glade desktop.
- 11.3 Ensure the GTK+ Basic button is selected on the Palette.
- 11.4 Click on the Button button then click on the second row of the vbox in window1.
- 11.5 Click on the Label button then click on the third row of the vbox in window1.
- 11.6 Window1 should now look like this.



- 11.7 We now need to adjust the properties of the new button and label to tidy the display.
- 11.8 Click on the button you just added to ensure its properties are displayed in the Properties window.
- 11.9 Click on the Widgets tab of the Properties window.
- 11.10 Change Name: to btnNumberTwo.
- 11.11 Change Label: to Number Two.
- 11.12 Click on the Packing tab.

- 11.13 Change Expand to Yes.
- 11.14 Change Fill to Yes.
- 11.15 Click on the Signals tab.
- 11.16 Click the button labelled ... on the signal property.
- 11.17 select Clicked under the heading GTKButton signals then click the OK button.
- 11.17 Click the Add button on the Signals tab to add the signal to the list.
- 11.18 Click on the label you added to window1 to ensure its properties are displayed in the property window.
- 11.19 Click on the Widgets tab of the Properties window.
- 11.20 Change Name: to lblOutput.
- 11.21 Erase the contents of the Label: field.
- 11.22 Click on the Save button of the Glade Main window.
- 11.23 Our GUI is now complete, all we need to do is test the application one last time before we add the final code.

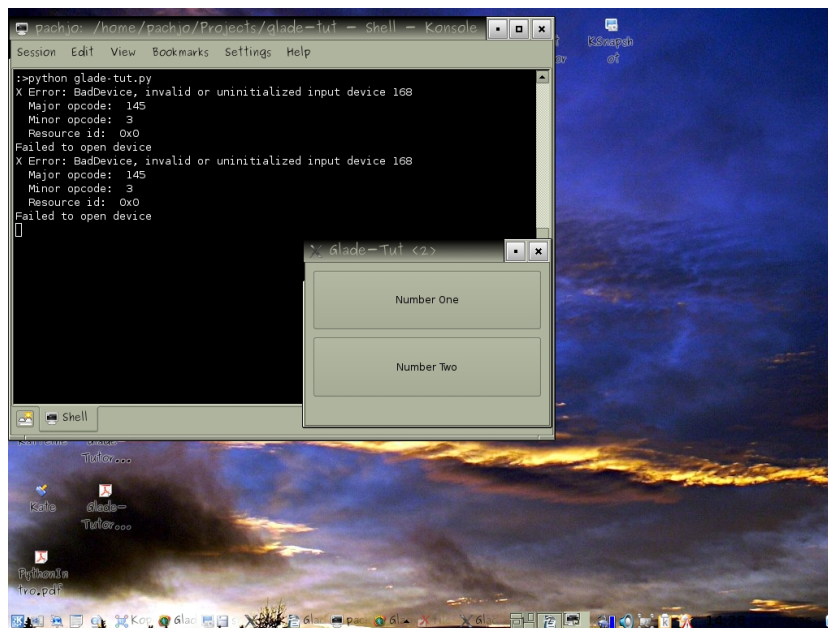
Python Run #5

12.1 select the console desktop and click in the console window.

12.2 Type the following command to run our application.

```
python glade-tut.py
```

12.3 Our window should now look like this.



12.4 We now have two nicely placed buttons which as yet do nothing.

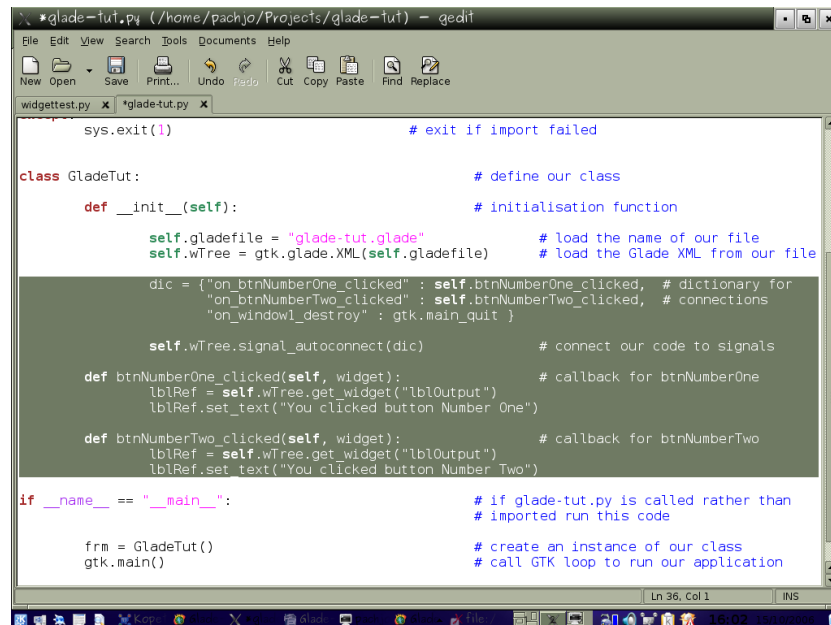
12.5 The label we placed on window1 is not visible as we erased its contents.

12.6 All we need to do now is to add some code to connect the signals from the two buttons to some callback functions to display some text in the label when we click them.

Gedit Complete The Python Code

13.1 select the Gedit desktop and select the Gedit window.

13.2 Amend the code to look like this.



```

*glade-tut.py (/home/pachjo/Projects/glade-tut) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
widgettest.py *glade-tut.py
sys.exit(1) # exit if import failed

class GladeTut: # define our class

    def __init__(self): # initialisation function

        self.gladefile = "glade-tut.glade" # load the name of our file
        self.wTree = gtk.glade.XML(self.gladefile) # load the Glade XML from our file

        dic = {"on_btnNumberOne_clicked" : self.btnNumberOne_clicked, # dictionary for
               "on_btnNumberTwo_clicked" : self.btnNumberTwo_clicked, # connections
               "on_window1_destroy" : gtk.main_quit }

        self.wTree.signal_autoconnect(dic) # connect our code to signals

    def btnNumberOne_clicked(self, widget): # callback for btnNumberOne
        lblRef = self.wTree.get_widget("lblOutput")
        lblRef.set_text("You clicked button Number One")

    def btnNumberTwo_clicked(self, widget): # callback for btnNumberTwo
        lblRef = self.wTree.get_widget("lblOutput")
        lblRef.set_text("You clicked button Number Two")

if __name__ == "__main__": # if glade-tut.py is called rather than
                           # imported run this code

    frm = GladeTut() # create an instance of our class
    gtk.main() # call GTK loop to run our application
Ln 36, Col 1 INS

```

13.3 We amend the dic= line to include the references to the callback functions of the two buttons.

13.4 We then add the two callback functions.

13.4.1 The callback function for btnNumberOne first gets a reference to the label lblOutput.

13.4.2 Then using this reference the text of the label is set to 'You clicked button Number One'.

13.4.3 The callback for btnNumberTwo is setup in the same way as the one for btnNumberOne.

13.5 That's all, our application is complete, click the Save button.

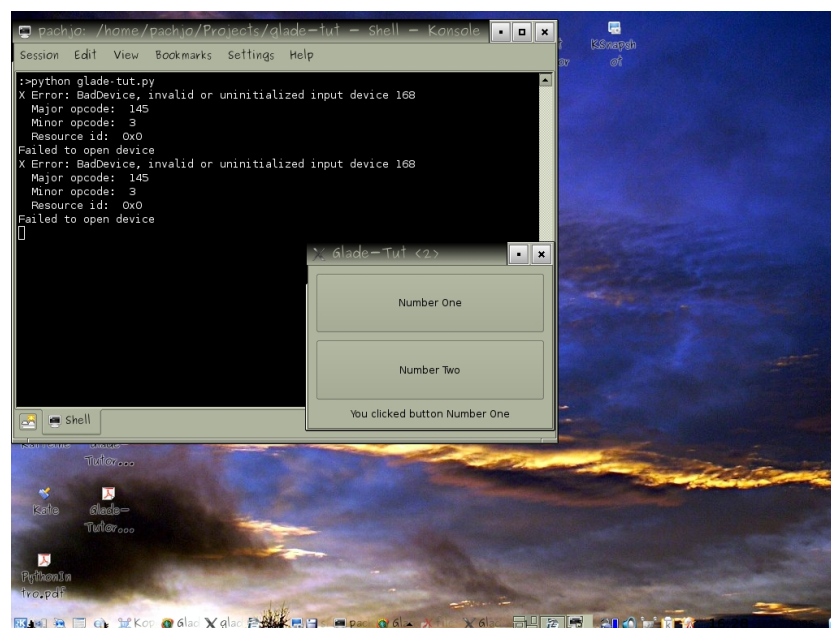
Python Run #6 The Finished Application

14.1 Select the console desktop and click in the console window.

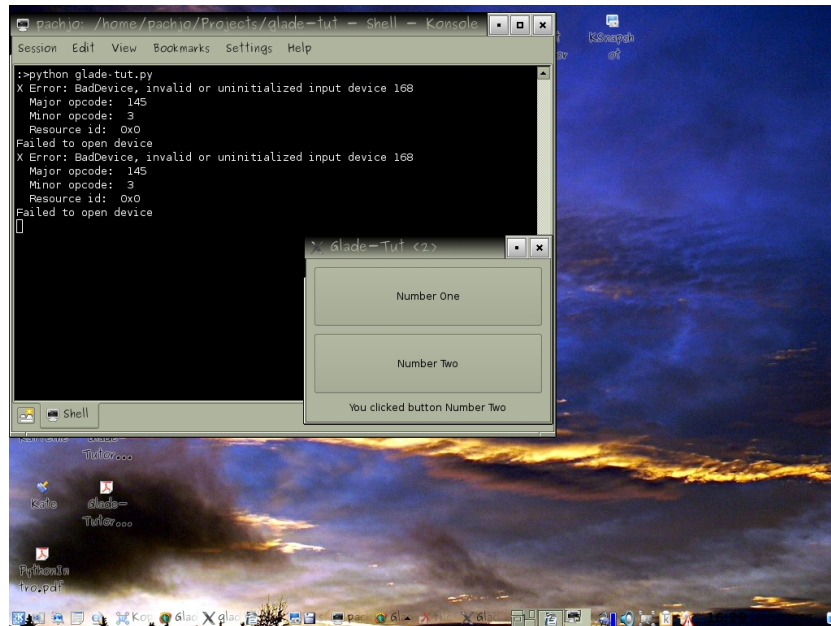
14.2 Type the following command to run our application.

```
python glade-tut.py
```

14.3 Our window should now look like this after you click button Number One.



14.4 And this is what it should like after clicking button Number Two.



14.5 So there you have it, a simply GUI application which hopefully will help you on your way to experiment more with Glade, Gedit and Python.

14.6 Refer to section 15 for some useful links.

Final Words

I hope this tutorial has helped you to understand the basics of how to start creating a GUI using Glade, Gedit and Python.

I would appreciate feedback on how you get on as well as any thoughts on the format of the guide itself.

I will use this information to assist me when creating more guides as I work my way through creating GUI applications in Linux.

Here are some useful links which might help you along the way:

<http://heather.cs.ucdavis.edu/~matloff/python.html>

<http://python-forum.org/py/index.php>

<http://www.pygtk.org/pygtk2reference/>

<http://glade.gnome.org/>

<http://www.python.org/>

Hope you enjoyed it.