sudo apt-get install glade-gnome

GTK

GTK+ is a toolkit, or a collection of libraries, which developers can use to develop GUI applications for Linux, OSX, Windows, and any other platform on which GTK+ is available.

It can be thought of in the same terms as MFC or the Win32 API on Windows, Swing and SWT in Java.

GTK written in C

GTK is based on 3 libraries   Glib, Pango and ATK

Glib wraps most of the C standard library functions.

GTK+ implement an Object oriented approach using Gobject.
Every piece of GTK+ GUI is comprised of one or more widgets which are objects.
All widgets will be derived from a base widget called GtkWidget.
eg.   Application window is a GtkWindow. Toolbar within the window is GtkToolbar

```
GObject
       +----GInitiallyUnowned
                      +----GtkObject
                                     +----GtkWidget
                                                    +----GtkContainer
                                                                   +----GtkBin
                                                                                +----
GtkWindow
```

The reason this heirarchy is so important is because when you're looking for functions, properties, and signals for any particular widget, you need to realize that the functions, properties, and signals of it's parent objects apply to it as well.

References
        Devhelp which will be available as a package, contains API documentation.
        library.gnome.org/devel/references

Naming Convension

The functions which manipulate these objects are in lower-case with underscores for spaces.
For example, gtk_window_set_title() is a function to set the title property of a GtkWindow object.

Introduction to Glade 3

Glade is a Rapid Application Development tool for desigining GTK+ applications.
Glade is a GTK+ application itself.
Helps in laying out the applications, glade file is a XML which describes hierarchy of the widgets comprising the interface.

Glade orginally generated C code to build GUI later discouraged and libglade generated at runtime.

Palette of GtkWidget s which can be used to build the application.
Inspector   shows your design as a tree

Property 4 tabs

General
properties for a GtkWidget

Packing

homogeneous: A property of the container widget which when set, tells GTK+ to allocate the same amount of space for each child.
expand: A property of the child being packed specifying if it should recieve extra space when the parent grows.
for eg
window with a menu. menu is child if you put expand “yes”
when you maximise the window the child will also get extra space
ie the menu will also grow.

GtkScrolledwindow expand is yes or else when more contents added
it will not grow.
fill: A property of the child being packed specifying whether any extra space
should be given to the child or used as padding around the child
for eg menu textviewwith scrollbar and statusbar
if middle widget put “yes” then only it will be fit otherwise padding between neighbours will come into effect .

Example texteditor

Common

Also contains properties but inherited from the parent objects

Signals

Objects emit a "signal" when something that might be useful to the programmer happens. These are similiar to "events" from Visual Basic

important signals

destroy signal is emmitted whenever a GTK object is destroyed.
So destroy signal emitted for our GtkWindow

sample program printing name entered in the text box

```
import sys

try:
    import pygtk
    pygtk.require("2.0")
except:
    pass
try:
    import gtk
except:
    print("GTK Not Availible")
    sys.exit(1)

class euca:
    def __init__(self):
        self.glade = "gtkbuilder.glade"
        self.builder = gtk.Builder()
        self.builder.add_from_file(self.glade)
        self.window = self.builder.get_object("window1")
        dic = {"on_button1_clicked":self.displaymsg}
        self.builder.connect_signals(dic)

    def displaymsg(self,widget):
        print self.builder.get_object("name").get_text()

if __name__ == "__main__":
    e = euca()
    gtk.main()
```

Notebook

Displaying notebook
 Inorder to display the notebook through the pygtk displaying tab should have something
 Other wise it will show some error

Adding new tab

Right click on the tab two options are there insert before and insert after

Changing the positon of the tabs
 Right click on the tab and select Edit tab seperately in the packing tab you can see the Position combo box put the position there . Exchange positions if already occupied

Text box

Add text box into window and one button on clicking button should get the text value entered

textbox	
name	tb_name

```
print "Text box value ",self.builder.get_object("tb_name").get_text()
```

ComboBox

Drag and drop a combobox

Name	combobox1

Appending values into the combobox

```
try:
    import pygtk
    pygtk.require("2.0")
    import gobject                    <--------
```

```
except:
    pass
```

add the following lines in __init__ so that combobox will be loaded with these values

```
        cbox = self.builder.get_object("combobox1")
      store = gtk.ListStore(gobject.TYPE_STRING)
      store.append(["vishal"])
      store.append(["arun"])
      store.append(["sabin"])
      cbox.set_model(store)
      cbox.append_text("hai")
      cell = gtk.CellRendererText()
      cbox.pack_start(cell, True)
      cbox.add_attribute(cell, 'text', 0)

      cbox.set_active(0)
```

Taking the selected text from combobox
add these lines in a button click

```
        cbox = self.builder.get_object("combobox1")
      model = cbox.get_model()
      active = cbox.get_active()
      print "value selected",model[active][0]
```

Treeview

1.Create a List store
TreeModel > Liststore   under objects it will create a new GtkListstore

Select the newly added liststore, Under the General tab you will find

Add or remove columns
Add or remove rows

1. firstly add the columns
you have to specify the datatype and Name
2 Add the content if you want from add or remove rows

2. Add the treeview from Control and Display

Add the treeview into the window it will ask for the Treeview model
add the model that you created in the liststore

Position the treeview and adjust height and width.

select the treeview and click edit
Go to hierarchy tab -- Add the columns there
Adding renderers
For each column added right click and select add child Text item
Then click on each renderer and show up the corresponding lisstore coloumn by changing
the Text property to lisstore corresponding index.

This will show up the added data in liststore

```
   act= [('jaos', 'mva'), ('sig', 'new')]
  import gobject

  #adding more values to tree view
st = gtk.ListStore(gobject.TYPE_STRING,gobject.TYPE_STRING)
st.append([act[0][0],act[0][1]])
st.append([act[1][0],act[1][1]])
ctree =  self.builder.get_object("treeview")
ctree.set_model(st)


  #displaying value selected
  def displaytree(self,widget,row,col):
model = widget.get_model()
text = model[row][0] + ", " + model[row][1]
print text
```

Toolbar

Firstly drag and drop the toolbar into the window

Container> Toolbar

Go to hierarchy you can add buttons into the toolbar there .

Specify the type
specify the Edit image
icon name having some icons select any one of them

signals specify
is at the bottom