MANUAL PYTHON-GLADE CON EJEMPLOS

El tutorial esta dividido en dos partes, en cada parte se explica como crear una ventana que tiene un texto (una etiqueta de texto para ser exactos), un cuadro donde escribir y un botón. La idea es que al escribir un texto en la entrada de texto (el cuadro), y al presionar el botón OK (o la tecla ENTER) el texto se muestra en la etiqueta de la forma "Hola **nuestro_texto**".

En la primer parte explica como crear el ejemplo directamente sobre el código, o sea construir la interfaz y conectarla directamente, todo esto desde el mismo archivo en python, o sea todo hecho a mano. Mientras que en la segunda parte se explica como crear la interfaz desde el diseñador de interfaces Glade (de forma separada y bastante intuitiva) y luego conectarla (llamarla) desde el archivo en python.

Pero antes, hay que instalar pygtk y el soporte para glade, desde debian (y derivadas como ubuntu) es tan fácil como hacer un:

 # apt-get install python-gtk2 python-glade2

Además para la segunda parte se necesita glade para crear la interfaz gráfica (Aunque no es necesario para ejecutarla), así que hacemos:

 # apt-get install glade

- **Primer Ejemplo: Directamente desde el código**

En este caso es recomendable echar una ojeada rápida al manual de pygtk, no es completamente necesario pero ayuda a entender mejor como funcionan los widgets y las señales.

A continuación el ejemplo, los comentarios explican el código:

```
01 #! /usr/bin/env python
02 # -*- coding: UTF-8 -*-
03
04 # Importamos el módulo pygtk y le indicamos que use la versión 2 (en caso de
05 # que existan varias versiones de pygtk instaladas en el sistema)
06 import pygtk
07 pygtk.require("2.0")
08
09 # Luego importamos el módulo de gtk para poder acceder a los controles de Gtk+
10 import gtk
11
12 # Creamos una clase que contenga la ventana principal del programa y los
13 # métodos de cada una las señales
14 class MainWin:
15
16     # Primero definimos como sera la ventana:
17     def __init__(self):
18
19         # Creamos una ventana toplevel (o sea que esta al frente de todas las
20         # ventanas) llamada "main_win" y fijamos su titulo como "Ejemplo 1"
21         main_win = gtk.Window(gtk.WINDOW_TOPLEVEL)
22         main_win.set_title("Ejemplo 1")
23
24         # A main_win le conectamos una señal (destroy), esto hará que cada
25         # vez que se presione el botón salir (la cruz del manejador de
26         # ventanas) se llamará al método on_quit que cerrara la ventana
```

```
27        main_win.connect("destroy", self.on_quit)
28
29        # Para agregar widgets (controles como botones, etiquetas, etc.) a la
30        # ventana, primero es necesario crear contenedores como cajas que
31        # contengan las widgets. En este ejemplo creamos una caja vertical con
32        # un espacio entre widgets de 5px y con la propiedad homogéneo en False
33        vbox = gtk.VBox(False, 5)
34
35        # Creamos una etiqueta con el texto "Hola mundo", se usa la palabra
36        # reservada "self" de python para poder hacer referencia a esta
37        # etiqueta desde otros métodos
38        self.label = gtk.Label("Hola mundo")
39
40        # Creamos un cuadro donde escribir (una entrada de texto en blanco)
41        # y luego le conectamos la señal "activate" que llama al método
42        # "on_button1_clicked", esto producirá que cuando se haga click en el
43        # botón Ok (que se creara mas adelante) la entrada de texto reaccione
44        self.entry = gtk.Entry()
45        self.entry.connect("activate", self.on_button1_clicked)
46
47        # Ahora creamos el botón, que sera el botón OK del inventario de
48        # botones de GNOME.
49        # Y luego le indicamos al botón que cuando le hagan click emita la
50        # señal "clicked" que llamará a "on_button1_clicked"
51        button = gtk.Button(stock=gtk.STOCK_OK)
52        button.connect("clicked", self.on_button1_clicked)
53
54        # Ahora que ya creamos las widgets (la etiqueta, la entrada de texto y
55        # el botón) hay que añadirlos a la caja vertical creada anteriormente
56
57        # Primero le añadimos la etiqueta llamada label a la caja vertical
58        vbox.add(self.label)
59
60        # Luego añadimos al inicio de la segunda fila la entrada de texto
61        # activando las propiedades de expandir, rellenar y espaciado.
62        vbox.pack_start(self.entry, True, True, 0)
63
64        # Finalmente en la tercer fila agregamos el botón.
65        vbox.pack_start(button, False, False, 0)
66
67        # Ahora agregamos la caja vertical a la ventana y luego se muestra
68        # la caja (y todo lo que contiene) en la ventana principal.
69        main_win.add(vbox)
70        main_win.show_all()
71
72
73    # Ahora dentro de nuestra clase principal "MainWin" tenemos que definir
```

```
74    # que hacen cada uno de los métodos que se llamaron anteriormente
75
76    # Primero definamos el método "on_button1_clicked"
77    def on_button1_clicked(self, widget):
78        # Primero obtenemos el texto que se escriba en la entrada de texto
79        texto = self.entry.get_text()
80        # Luego fijamos ese texto a la etiqueta de la forma "Hola texto".
81        self.label.set_text("Hola %s" % texto)
82
83    # Ahora se define el método "on_quit" que destruye la aplicación
84    def on_quit(self, widget):
85        gtk.main_quit()
86
87
88 # Para terminar iniciamos el programa
89 if __name__ == "__main__":
90    # Iniciamos la clase.
91    MainWin()
92    # Además iniciamos el método gtk.main, que genera un ciclo que se utiliza
93    # para recibir todas las señales emitidas por los botones y demás widgets.
94    gtk.main()
```

Como todos los script de python los podemos ejecutar haciendo:

$ python ejemplo.py

- **Segundo Ejemplo: pyGtk + Glade**

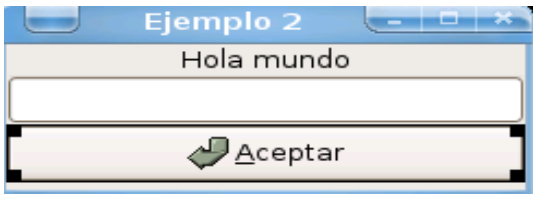
En este caso vamos a crear toda la interfaz gráfica usando glade, el cual crea un archivo con extensión ****.glade*** en donde guardan los nombres, señales, etc. de cada uno de los widget y de las ventanas. Así en el archivo .py solo se indica las widgets que se van a utilizar y que hacen los métodos.

Primero lanzamos glade y desde ventana principal crearemos un nuevo proyecto de gtk. Si se fijan bien el programa tiene una paleta con los widgets que se pueden usar y una ventana de propiedades que muestra la información de cada widget y permite modificarla (ambas se ven la imagen a continuación), estas lasusaremos frecuentemente así que ubiquenla (en glade2 se pueden activar/desactivar desde el menú ver)

Ahora desde la paleta hacemos click en el widget ventana (gtk.Window) para crear una ventana, luego seleccionamos la ventana creada y modificamos su información usando la ventana de propiedades en donde podemos cambiar su nombre, titulo, ancho, largo, etc. en este caso le cambiaremos el titulo a “Ejemplo 2″

El siguiente paso es crear la caja vertical, para eso en la paleta hacemos click en el widget caja vertical (gtk.VBox), luego otro click en la ventana en donde queremos colocarla y la dejamos con 3 filas. Nos queda algo como esto:

Continuaremos insertando una etiqueta (gtk.Label) en la primer fila, después insertamos una entrada de texto (gtk.Entry) en la segunda fila y por último un botón (gtk.Button) en la tercera fila. Así nos queda una ventana:



Ahora cambiamos las propiedades de esta etiqueta para que el nombre sera “label1″ y el texto que muestra lo cambiamos de “label1″ a “Hola mundo”. En el caso de la entrada de texto (gtk.entry) le dejamos su nombre como “entry1″ y para el botón le dejamos el nombre como “button1″ y escogemos el botón de inventario de aceptar(gtk-ok)

El próximo paso es conectar las señales que emiten los eventos de cada widget, esto se hace en propiedades, en la pestaña “Señales”. Primero seleccionamos la ventana (window1), buscamos la señal “destroy” y seleccionamos el manejador gtk_main_quit (creo que ya saben para que es esto).

De igual manera seleccionamos para el con el entry1 la señal “activate” y el manejador “on_button1_clicked”(si no lo encuentran escribanlo). Por último el para el button1 escogemos la señal “clicked” y el manejador “on_button1_clicked”

Nota: Se puedes usar manejadores personalizados, pero hay que tener cuidado el escribir el script en python

Finalmente hay que guardar el archivo con extensión .glade en el mismo directorio donde vayamos a crear el archivo de python, en mi caso lo llamare “GladeEjemplo.glade” (aunque tal como lo señalo mas adelante, se puede usar otros directorios si se tiene cuidado).

Por ultimo nuestro script queda así (los comentarios explican el código):

```
1  #! /usr/bin/env python
02 # -*- coding: UTF-8 -*-
03
04 # Importamos el módulo pygtk y le indicamos que use la versión 2
05 import pygtk
06 pygtk.require("2.0")
07
08 # Luego importamos el módulo de gtk y el gtk.glade, este ultimo que nos sirve
09 # para poder llamar/utilizar al archivo de glade
10 import gtk
11 import gtk.glade
12
13 # Creamos la clase de la ventana principal del programa
14 class MainWin:
15
16     def __init__(self):
17         # Le decimos a nuestro programa que archivo de glade usar (puede tener
18         # un nombre distinto del script). Si no esta en el mismo directorio del
19         # script habría que indicarle la ruta completa en donde se encuentra
20         self.widgets = gtk.glade.XML("GladeEjemplo.glade")
21
22         # Creamos un pequeño diccionario que contiene las señales definidas en
23         # glade y su respectivo método (o llamada)
24         signals = { "on_entry1_activate" : self.on_button1_clicked,
25                     "on_button1_clicked" : self.on_button1_clicked,
26                     "gtk_main_quit" : gtk.main_quit }
27
28         # Luego se auto-conectan las señales.
29         self.widgets.signal_autoconnect(signals)
30         # Nota: Otra forma de hacerlo es No crear el diccionario signals y
31         # solo usar "self.widgets.signal_autoconnect(self)" -->Ojo con el self
32
33         # Ahora obtenemos del archivo glade los widgets que vamos a
34         # utilizar (en este caso son label1 y entry1)
35         self.label1 = self.widgets.get_widget("label1")
36         self.entry1 = self.widgets.get_widget("entry1")
37
38     # Se definen los métodos, en este caso señales como "destroy" ya fueron
39     # definidas en el .glade, así solo se necesita definir "on_button1_clicked"
```

```
40    def on_button1_clicked(self, widget):
41        texto = self.entry1.get_text()
42        self.label1.set_text("Hola %s" % texto)
43
44  # Para terminar iniciamos el
    programa
45  if __name__ == "__main__":
46    MainWin()
47    gtk.main()
```

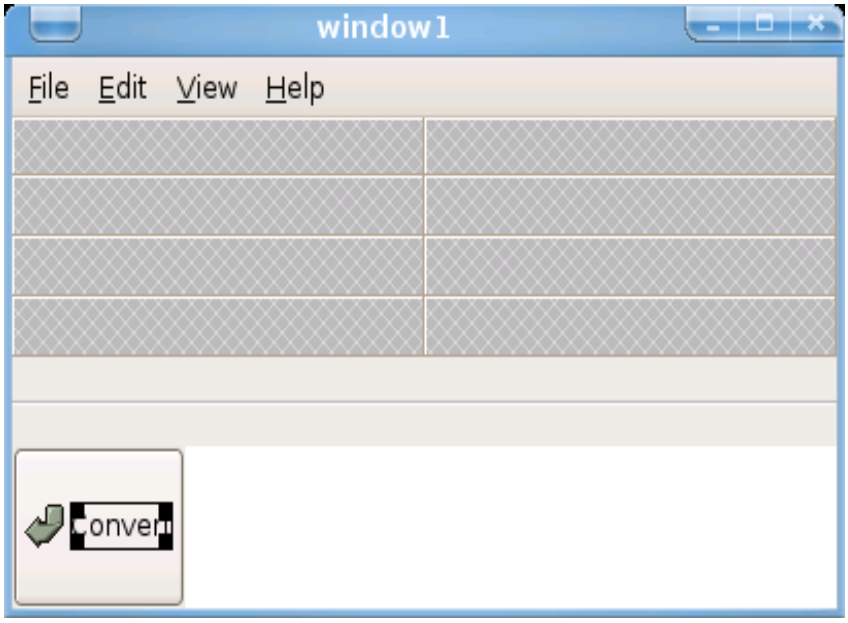
Vamos a crear una aplicación simple usando pygtk y glade, la aplicación que voy a crear es un programa que convierte unidades de temperatura (por ejemplo pasa de Celsius a Fahrenheit o Kelvin), claro que no es un gran programa pero es mucho mejor que el "hola mundo" que hice para el primer tutorial, la introducción a pygtk+glade .

En cuanto al nombre del programa, lo voy a llamar "python temperature converter" (pytemp para abreviar). Así que sin mas preámbulo pasemos al tutorial

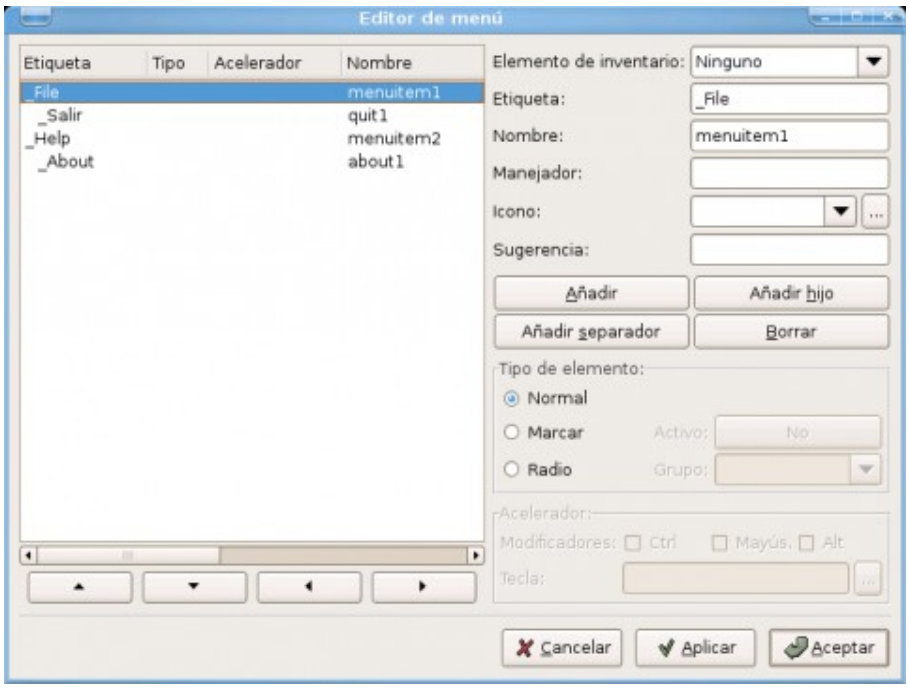
1. **Primera Etapa: Crear la ventana**

Desde glade cree un nuevo proyecto de gtk (que llame pytemp), una vez creado el proyecto, lo siguiente que creo es una nueva ventana (llamada window1 y que tenga el titulo "Temperature Converter") y en esta ventana agrego una caja vertical con 4 filas.

En la primera fila le pongo una barra de menú (que mas adelante voy a editar), en la segunda fila de la caja vertical agrego una tabla (de 2 columnas y 4 filas), en la tercera fila de la caja vertical pongo un separador horizontal y en la cuarta fila de la caja vertical creo una caja horizontal con dos columnas (dentro de la caja horizontal en la izquierda pongo un botón con la etiqueta "Convert" y que el botón use el icono gtk-ok, mientras que en la parte derecha de la caja horizontal pongo un textview). Se crea una ventana como la siguiente:



Luego editamos el menú superior, para ello seleccionamos la barra del menú y en las propiedades apretamos el botón "Editar menús…", nos sale una ventana como esta:



En esta ventana borramos todos los menús que no usaremos, así en en file solo dejamos la opción salir y en help solo dejamos la opción about (tal como lo muestra la imagen anterior)

Volviendo a la muestra ventana, ahora vamos a trabajar sobre la tabla, en la primera fila primera columna coloco una etiqueta con el texto “Value”.

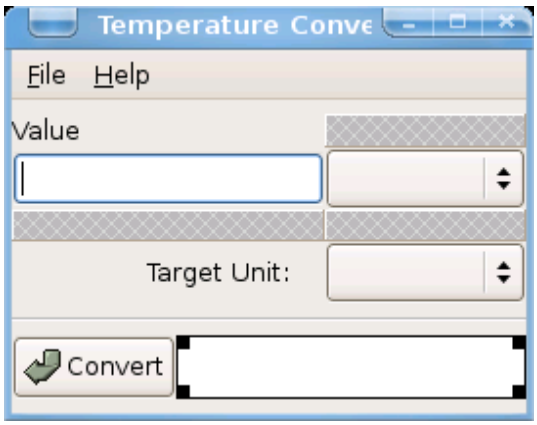
Luego en la segunda fila, en la primera columna agrego una entrada de texto que se llamara “entry1″ y en la segunda columna de la tabla coloco un GtkComboBox (que es una lista desplegable) llamado “combobox1″ que tenga los elementos Celsius, Fahrenheit y Kelvin (Nota: poner un elemento por linea)

> *La diferencia principal entre un ComboBox y un ComboBoxEntry es que si bien es cierto que ambos le permiten al usuario seleccionar algún item a partir de la una lista desplegable, el ComboBoxEntry también le permite al usuario ingresar un nuevo ítem, cosa que no permite el ComboBox, además de algunas diferencias en como se cargan/manejan*

En cuanto a la tercera fila, la dejo en blanco para que quede algo de separación entre los elementos

Luego en la cuarta fila, en la primera columna ubico una etiqueta con el texto “Target Unit:”, y le modifico la propiedad Alineación X dejándola en 0.80 (esto sirve para desplazar la etiqueta, 0.00 es completamente a la izquierda, 0.50 en el centro y 1.00 completamente a la derecha). En cuanto a la segunda columna coloco un GtkComboBox llamado “combobox2″ que tiene los mismos elementos que el anterior (el “combobox1″)

Si todo sale bien queda una ventana como esta:



Se crean las señales correspondientes que son las siguientes (ya explique como agragarlas):

- Elemento: “window1″, señal “destroy” y el manejador “gtk_main_quit”
- Elemento: “on_”button1″, señal “clicked” y el manejador “on_button1_clicked”
- Elemento: “quit1″ (En el menú File es la opción Quit), señal “activate” y el manejador “gtk_main_quit”
- Elemento: “about1″ (En el menú Help es la opción About), señal “activate” y el manejador “on_about1_activate”

2. Creando el código

Voy a tratar de explicar de una manera que se entienda claramente, lo cual no es necesariamente el orden que sigue el código, por lo que de vez en cuando se darán algunos saltos hacia atrás o adelante en el código. En todo caso el archivo final esta bien comentado, así que no es muy difícil que se pierdan.

Lo primero es importar los módulos que necesitamos (en este caso solo son pygtk, gtk y gtk.glade), para lo cual hacemos:

```
01 # Importamos los módulos necesarios
02 try:
03     import pygtk
04     pygtk.require('2.0') # Intenta usar la versión2
05 except:
06     # Algunas distribuciones vienen con GTK2, pero no con pyGTK (o pyGTKv2)
07     pass
08
09 try:
10     import gtk
11     import gtk.glade
12 except:
13     print "You need to install pyGTK or GTKv2 or set your PYTHONPATH correctly"
14     sys.exit(1)
```

*Ahora definimos varias funcion*es que convierten la temperatura. Para eso vamos a la wikipedia para saber que las conversiones básicas que son: C=(F-32)*(5/9) y K=C+273.15

Así nuestras funciones para convertir las temperaturas quedan:

```
01 # Definimos varias funciones que convierten la temperatura.
02
03 def fahrenheit2celsius(temp):
04     "Convert Fahrenheit to celsius"
05     celsius = (temp - 32) * 5.0 / 9.0
06     return celsius
07
08 def celsius2fahrenheit(temp):
09     "Convert Celsius to Fahrenheit"
10     fahrenheit = (9.0 / 5.0) * temp + 32
11     return fahrenheit
12
13 def kelvin2celsius(temp):
14     "Convert kelvin to Celsius"
15     celsius = temp - 273.15
```

```
    return celsius

def celsius2kelvin(temp):
    "Convert Celsius to kelvin"
    kelvin = temp + 273.15
    return kelvin
```

Nota: Solo cree 4 conversiones. Si se preguntan por ejemplo como pasar de Fahrenheit a Kelvin, simplemente pasan primero de Fahrenheit a Celsius y luego ese resultado (en Celsius) lo pasan a Kelvin

Luego vamos a crear una clase que almacena la información del programa, así solo tengo que escribir esta información una vez y luego la llamo cuando la necesite en alguna parte del programa, como por ejemplo en la ventana about (Acerca de)

```
# Creamos una clase que almacena la información del programa (después se usara)
class Info:
    "Store the program information"
    name = "pyTemp"
    version = "0.1"
    copyright = "Copyright © 2009 Daniel Fuentes B."
    authors = ["Daniel Fuentes Barría <dbfuentes @ gmail com>"]
    website = "http://pythonmania.wordpress.com/tutoriales/"
    description = "A temperature converter written in python using PyGTK"
    license = "This program is free software; you can redistribute itand/or \
modify it under the terms of the GNU General Public License as published by \
the Free Software Foundation; either version 2 of the License, or (at your \
option) any later version. \n\nThis program is distributed in the hope that \
it will be useful, but WITHOUT ANY WARRANTY; without even the implied \
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. \
See the GNU General Public License for more details. \n\nYou should have \
received a copy of the GNU General Public License along with this program; \
if not, write to the Free Software Foundation, Inc., 51 Franklin Street, \
Fifth Floor, Boston, MA 02110-1301, USA."
```

Fijense que la parte authors es una lista, así que si existieran mas de un autor, simplemente abría que agregarlos como mas elementos de la lista.

Ahora lo que creamos es una clase que tenga nuestra Interfaz gráfica, que es como sigue:

```
# Interfaz gráfica (gtk-glade), Clase para el Loop principal (de la GUI)
class MainGui:
    "GTK/Glade User interface. This is a pyGTK window"
    def __init__(self):
        # Le indicamos al programa que archivo XML de glade usar
        self.widgets = gtk.glade.XML("pytemp.glade")

        # se definen las signals
        signals = { "on_button1_clicked" : self.on_button1_clicked,
                    "on_about1_activate" : self.on_about1_activate,
                    "gtk_main_quit" : gtk.main_quit }
```

```
12
13          # y se autoconectan las signals.
14          self.widgets.signal_autoconnect(signals)
15
16          # Del archivo glade obtenemos los widgets a usar
17          self.entry1 = self.widgets.get_widget("entry1")
18          self.textview1 = self.widgets.get_widget("textview1")
19          self.combobox1 = self.widgets.get_widget("combobox1")
20          self.combobox2 = self.widgets.get_widget("combobox2")
21
22          # Para el ComboBox1 se fija por defecto la primera opción de la lista
23          self.combobox1.set_active(0)
24          # y en el ComboBox2 se fija por defecto la segunda opción de la lista
25          self.combobox2.set_active(1)
```

Esto fue explicado en el primer tutorial, asi que no voy a repetir la explicación, pero voy a hacer una observación respecto a los ComboBox.

Por defecto al cargar un ComboBox aparecen sin nada seleccionado, como no quiero que pase eso, le indico cual elemento tiene que estar activo por defecto (escogido inicialmente) usando el set_active(). En cuanto al orden de los elementos del ComboBox el primero tiene indice 0, el segundo 1 y así sucesivamente, por lo que le estoy diciendo al programa que el combobox1 por defecto tenga seleccionado “Celsius” y el combobox2 tenga “Fahrenheit”

Lo que sigue es crear un par de ventanas, la primera sera una ventana de error

```
01 # Ventana genérica de error (se le pasan los mensajes de error y los
02 # muestra en una ventana de dialogo)
03 def error(self, message):
04     "Display the error dialog "
05     dialog_error = gtk.MessageDialog(parent=None, flags=0, buttons=gtk.BUTTONS_OK)
06     dialog_error.set_title("Error")
07     label = gtk.Label(message)
08     dialog_error.vbox.pack_start(label, True, True, 0)
09     # Con show_all() mostramos el contenido del cuadro de dialogo (en este
10     # caso solo tiene la etiqueta) si no se hace el dialogo aparece vacío
11     dialog_error.show_all()
12     # El run y destroy hace que la ventana se cierre al apretar el botón
13     dialog_error.run()
14     dialog_error.destroy()
```

Esta ventana (que para ser exactos es un pequeño dialogo) nos muestra un mensaje de error personalizado, si se preguntan porqué no la cree en glade, es simple, si la hubiera hecho en glade para cada error tendría que estar modificando etiquetas y similares, mientras que de esta manera simplemente cuando quiero mostrar un mensaje hago self.error(“Mi mensaje”)..

La siguiente es la ventana acerca de

```
01 # Ventana About (conocida como Acerca de).
02 def about_info(self, data=None):
03     "Display the About dialog "
04     about = gtk.AboutDialog()
05     about.set_name(Info.name)
06     about.set_version(Info.version)
```

```
07    about.set_comments(Info.description)
08    about.set_copyright(Info.copyright)
09
10    def openHomePage(widget,url,url2): # Para abrir el sitio
11        import webbrowser
12        webbrowser.open_new(url)
13
14    gtk.about_dialog_set_url_hook(openHomePage,Info.website)
15    about.set_website(Info.website)
16    about.set_authors(Info.authors)
17    about.set_license(Info.license)
18    about.set_wrap_license(True) # Adapta el texto a la ventana
19    about.run()
20    about.destroy()
```

En este caso creo la ventana desde el código (y no desde glade) porque defino una función que al hacer click en la dirección del proyecto, la abre en el navegador web por defecto. Nota: se puede hacer lo mismo con las direcciones de email y similares, solo hay que revisar la documentación.

Ahora nos queda establecer las acciones que hay que realizar al hacer click en un botón o en el menú. En el caso del Help -> About es bastante sencillo, resultando así:

```
1  # Definimos la ventana about (help > About)
2  def on_about1_activate(self, widget):
3      "Open the About windows"
4      self.about_info()
```

En cuanto al botón para convertir las unidades la cosa se un poco mas complicada, ya que al apretarlo tiene que obtener el valor desde el entry1 y el texto activo (opción escogida) desde cada uno de los ComboBox.

En teoría haciendo un combobox.get_active() se obtiene el índice del valor, pero esto tiene sus inconvenientes: El primero es que el valor puede ser negativo, como por ejemplo -1 si no hay ninguna selección. El otro problema es que si por ejemplo tenemos ordenada una lista de cierta forma y en un futuro agregamos un elemento en medio de esa lista (por ejemplo que a una lista de ciudades ordenadas alfabéticamente le agregue una nueva ciudad) los idices de algunos elementos pueden quedar desplazados, produciendo un desastre en nuestro codigo

Por estos motivos es mejor obtener el texto activo que el indice, para lo cual vamos a usar una función que nos devuelva el texto activo de un combobox (para ser mas exactos vamos a usar una función propuesta en la documentación de pygtk).

```
1  # Función que obtiene el texto de la opción seleccionada en un ComboBox
2  # Se usa para obtener que unidad de temperatura seleccionada de la lista
3  def valor_combobox(combobox):
4      model = combobox.get_model()
5      activo = combobox.get_active()
6      if activo < 0:
7          return None
8      return model[activo][0]
```

Luego hacemos la conversion con esto:

```
01    # Definimos las acciones a realizar al apretar el botón de convertir
02    def on_button1_clicked(self, widget):
03        "Convert button"
04        # Se crea un buffer en donde se guardaran los resultados
```

```
05        text_buffer = gtk.TextBuffer()
06        # Se obtiene el valor para convertir desde la entrada
07        valor = self.entry1.get_text()
08        # Obtiene la opción escogida en los 2 ComboBoxs
09        selec1 = valor_combobox(self.combobox1)
10        selec2 = valor_combobox(self.combobox2)
11        try:
12            # Intenta transformar el valor ingresado en un numero. En caso
13            # de fallar (por ejemplo falla si lo ingresado son letras) se
14            # lanza la excepción, si es exitoso se continua con la conversión
15            temp_ini = float(valor)
16
17            # Inicia la conversión adecuada dependiendo de la opción
18            # escogida en los 2 ComboBoxs (selec1 y selec2)
19            if selec1 == "Celsius" and selec2 == "Fahrenheit":
20                text_buffer.set_text(str(celsius2fahrenheit(temp_ini)))
21            elif selec1 == "Celsius" and selec2 == "Kelvin":
22                text_buffer.set_text(str(celsius2kelvin(temp_ini)))
23            elif selec1 == "Fahrenheit" and selec2 == "Celsius":
24                text_buffer.set_text(str(fahrenheit2celsius(temp_ini)))
25            elif selec1 == "Fahrenheit" and selec2 == "Kelvin":
26                # Pasamos primero de F a Celsius, luego de Celsius a Kelvin
27                conversion1 = fahrenheit2celsius(temp_ini)
28                text_buffer.set_text(str(celsius2kelvin(conversion1)))
29            elif selec1 == "Kelvin" and selec2 == "Celsius":
30                text_buffer.set_text(str(kelvin2celsius(temp_ini)))
31            elif selec1 == "Kelvin" and selec2 == "Fahrenheit":
32                 # Pasamos primero de Kelvin a Celsius, luego de Celsius a F
33                 conversion1 = kelvin2celsius(temp_ini)
34                 text_buffer.set_text(str(celsius2fahrenheit(conversion1)))
35            else:
36                # Se produce cuando las dos selecciones son iguales
37                self.error("The initial and target units are the same")
38
39            # Luego se fija (muestra) el buffer (que contiene la temperatura
40            # convertida) en textview1 (el cuadro la lado del botón)
41            self.textview1.set_buffer(text_buffer)
42
43        except:
44            if (len(valor) == 0):
45                # Se produce si no se ingresa nada en la entry1
46                self.error("Please enter a value")
47            else:
48                # Se produce si no se ingresa un numero ( por ejemplo se
49                # ingresan letras o símbolos)
50                self.error("The value entered is not valid.\nEnter a \
51 valid number")
```

La idea principal detrás del try: es que que tome el valor ingresado en el “entry1″ y lo transforme en un numero flotante, si no lo puede transformar (porque no es un numero) se lanza la excepción y en caso de que si lo transforme continúe haciendo la transformación que corresponda

Finalmente agregamos un par de cosas (como que se inicie la clase MainGui) y el código completo:

```
001 #! /usr/bin/env python
002 # -*- coding: UTF-8 -*-
003
004 # pytemp - A temperature converter written in python using PyGTK
005 # Copyright (C) 2009 Daniel Fuentes Barría
006 #
007 # This program is free software; you can redistribute it and/or modify
008 # it under the terms of the GNU General Public License as published by
009 # the Free Software Foundation; either version 2 of the License, or
010 # (at your option) any later version.
011 #
012 # This program is distributed in the hope that it will be useful,
013 # but WITHOUT ANY WARRANTY; without even the implied warranty of
014 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
015 # GNU General Public License for more details.
016 #
017 # You should have received a copy of the GNU General Public License along
018 # with this program; if not, write to the Free Software Foundation, Inc.,
019 # 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
020
021 # Importamos los módulos necesarios
022 try:
023     import pygtk
024     pygtk.require('2.0') # Intenta usar la versión2
025 except:
026     # Algunas distribuciones vienen con GTK2, pero no con pyGTK (o pyGTKv2)
027     pass
028
029 try:
030     import gtk
031     import gtk.glade
032 except:
033     print "You need to install pyGTK or GTKv2 or set your PYTHONPATH correctly"
034     sys.exit(1)
035
036 # Definimos varias funciones que convierten la temperatura.
037
038 def fahrenheit2celsius(temp):
039     "Convert Fahrenheit to celsius"
040     celsius = (temp - 32) * 5.0 / 9.0
041     return celsius
042
043 def celsius2fahrenheit(temp):
044     "Convert Celsius to Fahrenheit"
045     fahrenheit = (9.0 / 5.0) * temp + 32
046     return fahrenheit
```

```
047
048 def kelvin2celsius(temp):
049     "Convert kelvin to Celsius"
050     celsius = temp - 273.15
051     return celsius
052
053 def celsius2kelvin(temp):
054     "Convert Celsius to kelvin"
055     kelvin = temp + 273.15
056     return kelvin
057
058 # Función que obtiene el texto de la opción seleccionada en un ComboBox
059 # Se usa para obtener que unidad de temperatura seleccionada de la lista
060 def valor_combobox(combobox):
061     model = combobox.get_model()
062     activo = combobox.get_active()
063     if activo < 0:
064         return None
065     return model[activo][0]
066
067 # Creamos una clase que almacena la información del programa (después se usara)
068 class Info:
069     "Store the program information"
070     name = "pyTemp"
071     version = "0.1"
072     copyright = "Copyright © 2009 Daniel Fuentes B."
073     authors = ["Daniel Fuentes Barría <dbfuentes @ gmail com>"]
074     website = "http://pythonmania.wordpress.com/tutoriales/"
075     description = "A temperature converter written in python using PyGTK"
076     license = "This program is free software; you can redistribute itand/or \
077 modify it under the terms of the GNU General Public License as published by \
078 the Free Software Foundation; either version 2 of the License, or (at your \
079 option) any later version. \n\nThis program is distributed in the hope that \
080 it will be useful, but WITHOUT ANY WARRANTY; without even the implied \
081 warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. \
082 See the GNU General Public License for more details. \n\nYou should have \
083 received a copy of the GNU General Public License along with this program; \
084 if not, write to the Free Software Foundation, Inc., 51 Franklin Street, \
085 Fifth Floor, Boston, MA 02110-1301, USA."
086
087
088 # Interfaz gráfica (gtk-glade), Clase para el Loop principal (de la GUI)
089 class MainGui:
090     "GTK/Glade User interface. This is a pyGTK window"
091     def __init__(self):
092         # Le indicamos al programa que archivo XML de glade usar
093         self.widgets = gtk.glade.XML("pytemp.glade")
094
095         # se definen las signals
096         signals = { "on_button1_clicked" : self.on_button1_clicked,
```

```
097                 "on_about1_activate" : self.on_about1_activate,
098                 "gtk_main_quit" : gtk.main_quit }
099
100         # y se autoconectan las signals.
101         self.widgets.signal_autoconnect(signals)
102
103         # Del archivo glade obtenemos los widgets a usar
104         self.entry1 = self.widgets.get_widget("entry1")
105         self.textview1 = self.widgets.get_widget("textview1")
106         self.combobox1 = self.widgets.get_widget("combobox1")
107         self.combobox2 = self.widgets.get_widget("combobox2")
108
109         # Para el ComboBox1 se fija por defecto la primera opción de la lista
110         self.combobox1.set_active(0)
111         # y en el ComboBox2 se fija por defecto la segunda opción de la lista
112         self.combobox2.set_active(1)
113
114     # A continuación se definen/crean las ventanas especiales (about, dialogo,
115     # etc) y las acciones a realizar en ellas.
116
117     # Ventana genérica de error (se le pasan los mensajes de error y los
118     # muestra en una ventana de dialogo)
119     def error(self, message):
120         "Display the error dialog "
121         dialog_error = gtk.MessageDialog(parent=None, flags=0, buttons=gtk.BUTTONS_OK)
122         dialog_error.set_title("Error")
123         label = gtk.Label(message)
124         dialog_error.vbox.pack_start(label, True, True, 0)
125         # Con show_all() mostramos el contenido del cuadro de dialogo (en este
126         # caso solo tiene la etiqueta) si no se hace el dialogo aparece vacío
127         dialog_error.show_all()
128         # El run y destroy hace que la ventana se cierre al apretar el botón
129         dialog_error.run()
130         dialog_error.destroy()
131
132     # Ventana About (conocida como Acerca de).
133     def about_info(self, data=None):
134         "Display the About dialog "
135         about = gtk.AboutDialog()
136         about.set_name(Info.name)
137         about.set_version(Info.version)
138         about.set_comments(Info.description)
139         about.set_copyright(Info.copyright)
140
141         def openHomePage(widget,url,url2): # Para abrir el sitio
142             import webbrowser
143             webbrowser.open_new(url)
144
145         gtk.about_dialog_set_url_hook(openHomePage,Info.website)
146         about.set_website(Info.website)
```

```
147         about.set_authors(Info.authors)
148         about.set_license(Info.license)
149         about.set_wrap_license(True) # Adapta el texto a la ventana
150         about.run()
151         about.destroy()
152 
153 
154     # Ahora declaramos las acciones a realizar (por menús, botones, etc.):
155 
156     # Definimos la ventana about (help > About)
157     def on_about1_activate(self, widget):
158         "Open the About windows"
159         self.about_info()
160 
161     # Definimos las acciones a realizar al apretar el botón de convertir
162     def on_button1_clicked(self, widget):
163         "Convert button"
164         # Se crea un buffer en donde se guardaran los resultados
165         text_buffer = gtk.TextBuffer()
166         # Se obtiene el valor para convertir desde la entrada
167         valor = self.entry1.get_text()
168         # Obtiene la opción escogida en los 2 ComboBoxs
169         selec1 = valor_combobox(self.combobox1)
170         selec2 = valor_combobox(self.combobox2)
171         try:
172             # Intenta transformar el valor ingresado en un numero. En caso
173             # de fallar (por ejemplo falla si lo ingresado son letras) se
174             # lanza la excepción, si es exitoso se continua con la conversión
175             temp_ini = float(valor)
176 
177             # Inicia la conversión adecuada dependiendo de la opción
178             # escogida en los 2 ComboBoxs (selec1 y selec2)
179             if selec1 == "Celsius" and selec2 == "Fahrenheit":
180                 text_buffer.set_text(str(celsius2fahrenheit(temp_ini)))
181             elif selec1 == "Celsius" and selec2 == "Kelvin":
182                 text_buffer.set_text(str(celsius2kelvin(temp_ini)))
183             elif selec1 == "Fahrenheit" and selec2 == "Celsius":
184                 text_buffer.set_text(str(fahrenheit2celsius(temp_ini)))
185             elif selec1 == "Fahrenheit" and selec2 == "Kelvin":
186                 # Pasamos primero de F a Celsius, luego de Celsius a Kelvin
187                 conversion1 = fahrenheit2celsius(temp_ini)
188                 text_buffer.set_text(str(celsius2kelvin(conversion1)))
189             elif selec1 == "Kelvin" and selec2 == "Celsius":
190                 text_buffer.set_text(str(kelvin2celsius(temp_ini)))
191             elif selec1 == "Kelvin" and selec2 == "Fahrenheit":
192                  # Pasamos primero de Kelvin a Celsius, luego de Celsius a F
193                  conversion1 = kelvin2celsius(temp_ini)
194                  text_buffer.set_text(str(celsius2fahrenheit(conversion1)))
195             else:
196                 # Se produce cuando las dos selecciones son iguales
```

```
197             self.error("The initial and target units are the same")
198
199         # Luego se fija (muestra) el buffer (que contiene la temperatura
200         # convertida) en textview1 (el cuadro la lado del botón)
201         self.textview1.set_buffer(text_buffer)
202
203     except:
204         if (len(valor) == 0):
205             # Se produce si no se ingresa nada en la entry1
206             self.error("Please enter a value")
207         else:
208             # Se produce si no se ingresa un numero ( por ejemplo se
209             # ingresan letras o símbolos)
210             self.error("The value entered is not valid.\nEnter a \
211 valid number")
212
213
214 if __name__== "__main__":
215     MainGui()
216     gtk.main()
```

Y así se ve muestro pequeño programa funcionando: