[GTK+ es una librería para la creación de interfaces gráficas de usuario](https://python-gtk-3-tutorial.readthedocs.io/en/latest/index.html). Su licencia permite desarrollar programas libres así como aplicaciones comerciales. Todo esto sin tener que pagar licencias o regalías para su uso.

Tenemos dos vías principales de acceso al diseño de interfaces con PyGTK. La primera de estas vías se basa en un editor de texto.

Más adelante una vez que nos hayamos familiarizado con los widgets que tenemos a nuestro alcance con GTK, utilizaremos una aplicación que nos facilitará la creación de nuestra interfaz.  
Como primer ejemplo para ver cómo funciona nos podemos remitir al siguiente código:

1. *# -\*- coding: utf8 -\*-*
2. **from** gi.repository **import** Gtk
3. win = Gtk.Window()
4. win.connect(**"delete-event"**, Gtk.main\_quit)
5. win.show\_all()
6. Gtk.main()

El ejemplo anterior es un mínimo. Crea un objeto que es una ventana , con una acción “Gtk.main\_quit” que te permite cerrar o destruir el objeto al producirse una señal de terminar, “delete-event”.

El siguiente ejemplo, es un “Hola mundo”, que muestra el saludo al hacer click en un botón:

1. *# -\*- coding: utf-8 -\*-*
2. **from** gi.repository **import** Gtk
3. **class** MyWindow(Gtk.Window):
4. **def** \_\_init\_\_(self):
5. Gtk.Window.\_\_init\_\_(self, title=**"Hola Mundo!"**)
6. self.button = Gtk.Button(label=**"Hazme click"**)
7. self.button.connect(**"clicked"**, self.on\_button\_clicked)
8. self.add(self.button)
9. **def** on\_button\_clicked(self, widget):
10. print(**"Hola Mundo!"**)
11. **def** main():
12. win = MyWindow()
13. win.connect(**"delete-event"**, Gtk.main\_quit)
14. win.show\_all()
15. Gtk.main()

La estructura de este ejercicio se basa en una clase llamada MyWindow y una pequeña función “main” donde se creará una instancia de la clase GUI.  
  
En el constructor de la clase MyWindow crearemos una ventana e incluiremos en ella un botón. Los métodos de la clase se asocian a las señales que se generan cuando la aplicación está en marcha. Se cierra la ventana, se pulsa un botón, etc …

Se puede observar en el código que al pulsar el botón se imprime el mensaje “Hola Mundo”. Cada señal o método lleva un bloque de código asociado donde se implementa la funcionalidad de cada objeto que está incluido en la ventana.

Veamos los comentarios específicos para cada línea:  
- L2: Necesitamos importar el módulo Gtk para tener acceso a todos los objetos y funciones que nos proporciona GTK+.  
- L5: Se crea la ventana vacía.  
- L[8-9]: Creamos un botón e indicamos que cuando se reciba la señal “clicked” se ejecute el método on\_button\_clicked.  
- L10: Es necesario añadir el botón a la ventana para trabajar sobre el. En este primer ejemplo simplemente lo añadimos a la ventana.  
- L17: Conectamos la señal Gtk.main\_quita nuestro evento"delete-event"

- L18: Método para mostrar la ventana y los objetos que contiene.  
- L19: Inicia el bucle de procesamiento de GTK+. Finaliza cuando la ventana se cierra, línea 18.  
  
**Señales**  
Como la mayoría de herramientas para creación de interfaces de usuario, GTK+ utiliza un modelo de programación orientado a eventos. Cuando el usuario no está haciendo nada, GTK + ejecuta un bucle principal y espera la llegada de eventos. Si el usuario realiza alguna acción - por ejemplo, un clic del ratón - a continuación, el bucle principal "despierta" y envía un evento para GTK+.

Cuando un widget-objeto recibe un evento, lo que ocurre es que genera una o varias señales. Las señales se encargan de notificar a nuestra aplicación que algo ha ocurrido. Esta señales están conectadas a funciones que se encargan de ejecutar bloques de código en función de la señal recibida.

A estas funciones se les llama comúnmente “callbacks”. Llamamos a estas funciones cuando queremos que ocurra alguna acción. Por ejemplo, cuando queremos que al hacer *clic* sobre un botón se hagan una serie de cálculos y se muestre un resultado. Cuando finaliza esta función, regresamos al bucle principal a la espera de nuevos eventos.

Veamos un ejemplo básico.  
id\_conexion = widget.connect(“evento”, función\_asociada, datos)

widget es una instancia de un objeto creado en nuestra aplicación. Mediante el método “connect” asociamos a esta instancia la señal “evento” con el bloque de código a ejecutar (cuando se reciba la señal) “función\_asociada”. Por último, el parámetro datos se utiliza para pasar algunos datos cuando ocurre la señal. Este argumento es completamente opcional.  
  
La función “connect” devuelve un identificador de la asociación entre evento y función asociada. Este identificador es necesario para desconectar el par evento-fución\_asociada. En el caso de que no se utilice durante un periodo concreto en la vida de la aplicación. La llamada sería:  
widget.disconnect(id\_conexion)  
  
En la mayoría de las aplicaciones se debe conectar en la ventana principal el evento “delete-event”. Este evento se genera cuando se cierra la ventana principal. El controlador (id) por defecto para esta señal, destruye la ventana, pero no termina la aplicación. La forma correcta de hacerlo es la siguiente:  
window.connect(“delete-event”, Gtk.main\_quit)

**Propiedades**  
Las propiedades nos indican la configuración y el estado de los widgets. Cada widget tiene su propio conjunto de propiedades concreto. Por ejemplo, un botón tiene la propiedad “label” que contiene el texto que se mostrará dentro del botón.

Las propiedades de cada widget se puede especificar cuando se crea una instancia del widget. Por decirlo de otra forma, llamando al constructor de la clase con los parámetros-propiedades que necesitemos para inicializar el widget. Veamos un ejemplo:

Etiqueta con el texto “Hola Mundo” alineado a la derecha con un ángulo de 25 grados:

mi\_etiqueta = Gtk.Label(label=”Hola Mundo”, angle=25, haling=Gtk.Align.END)  
lo cual es equivalente a:  
mi\_etiqueta = Gtk.Label() mi\_etiqueta.set\_label(“Hola Mundo”) mi\_etiqueta.set\_angle(25) mi\_etiqueta.set\_halign(Gtk.Align.END)  
  
para recuperar el texto de la etiqueta o cualquier otra propiedad:

texto = mi\_etiqueta.get\_label()   
  
**Widgets**

Cada pieza de una interfaz gráfica de usuario GTK+ se compone de uno o varios "widgets" que son objetos. Todos los widgets derivan de un widget de base llamado GtkWidget. Por ejemplo, una ventana de una aplicación es un widget llamado GtkWindow. La barra de herramientas dentro de la ventana es un widget llamado GtkToolbar.

Aunque un GtkWindow es también un GtkWidget, un GtkWidget no es necesariamente un GtkWindow. Los widgets hijos heredan de sus objetos padre para extender la funcionalidad del objeto. Se trata de programación orientada a objetos.

Podemos mirar en cualquier widget en la documentación de referencia de GTK + para ver qué objetos se derivan de él. En el caso de GtkWindow, se ve una estructura como esta:  
  
\* gobject.GObject  
 - gtk.Object  
 + gtk.Widget  
 \* gtk.Container  
 - gtk.Bin  
 + gtk.Window  
  
Como puedes ver, un objeto GtkWindow hereda de un objeto GtkBin que se deriva de GtkContainer, y así sucesivamente. La razón de que esta jerarquía sea tan importante es porque cuando estás buscando funciones, propiedades o señales de cualquier widget, es necesario darse cuenta de que las funciones, propiedades o las señales de los objetos padre se aplican también a ese widget hijo.

También empezamos a ver que surge una convención de nombres. Esto es bastante útil. Todos los objetos que comienzan con Gtk pertenecen a la librería GTK+. Todos los objetos (y por lo tanto Widgets) distinguen entre mayúsculas y minúsculas.

Las funciones que manipulan estos objetos se identifican en minúsculas con guiones para espacios. Por ejemplo, gtk\_window\_set\_title() es una función para establecer la propiedad de título de un objeto GtkWindow.

Dejamos un ejemplo simple de uso del widget ProgressBar (una barra de progreso para mostrar el completado de diferentes acciones) haciendo uso de lo ya mencionado en el tema del curso.  
  
*# -\*- coding: utf-8 -\*-*

**from** gi.repository **import** Gtk, GObject

**class** ProgressBarWindow(Gtk.Window):

**def** \_\_init\_\_(self):

Gtk.Window.\_\_init\_\_(self, title=**"Demo del Widget ProgressBar"**)

self.set\_border\_width(10)

vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)

self.add(vbox)

self.progressbar = Gtk.ProgressBar()

vbox.pack\_start(self.progressbar, **True**, **True**, 0)

button = Gtk.CheckButton(**"Mostrar completado"**)

button.connect(**"toggled"**, self.on\_show\_text\_toggled)

vbox.pack\_start(button, **True**, **True**, 0)

button = Gtk.CheckButton(**"Modo activo"**)

button.connect(**"toggled"**, self.on\_activity\_mode\_toggled)

vbox.pack\_start(button, **True**, **True**, 0)

button = Gtk.CheckButton(**"Completado inverso"**)

button.connect(**"toggled"**, self.on\_right\_to\_left\_toggled)

vbox.pack\_start(button, **True**, **True**, 0)

self.timeout\_id = GObject.timeout\_add(50, self.on\_timeout, **None**)

self.activity\_mode = **False**

**def** on\_show\_text\_toggled(self, button):

show\_text = button.get\_active()

**if** show\_text:

text = **"Auto Completado "**+str(self.progressbar.get\_fraction()\*100)

**else**:

text = **None**

self.progressbar.set\_text(text)

self.progressbar.set\_show\_text(show\_text)

**def** on\_activity\_mode\_toggled(self, button):

self.activity\_mode = button.get\_active()

**if** self.activity\_mode:

self.progressbar.pulse()

**else**:

self.progressbar.set\_fraction(0.0)

**def** on\_right\_to\_left\_toggled(self, button):

value = button.get\_active()

self.progressbar.set\_inverted(value)

**def** on\_timeout(self, user\_data):

*"""*

*Actualizamos el valor de la barra de progreso*

*"""*

**if** self.activity\_mode:

self.progressbar.pulse()

**else**:

new\_value = self.progressbar.get\_fraction() + 0.01

**if** new\_value > 1:

new\_value = 0

self.progressbar.set\_fraction(new\_value)

self.progressbar.set\_text(str(self.progressbar.get\_fraction()\*100))

*# Como es una funcion timeout, devuelve True para asi*

*# continuar llamándose*

**return True**

win = ProgressBarWindow()

win.connect(**"delete-event"**, Gtk.main\_quit)

win.show\_all()

Gtk.main()

<https://python-gtk-3-tutorial.readthedocs.io/en/latest/label.html>