

Interfaz Gráfica de Usuario

PROGRAMACIÓN EN
Python

por
Víctor Melchor

Interfaces gráficas de usuario

Concepto:

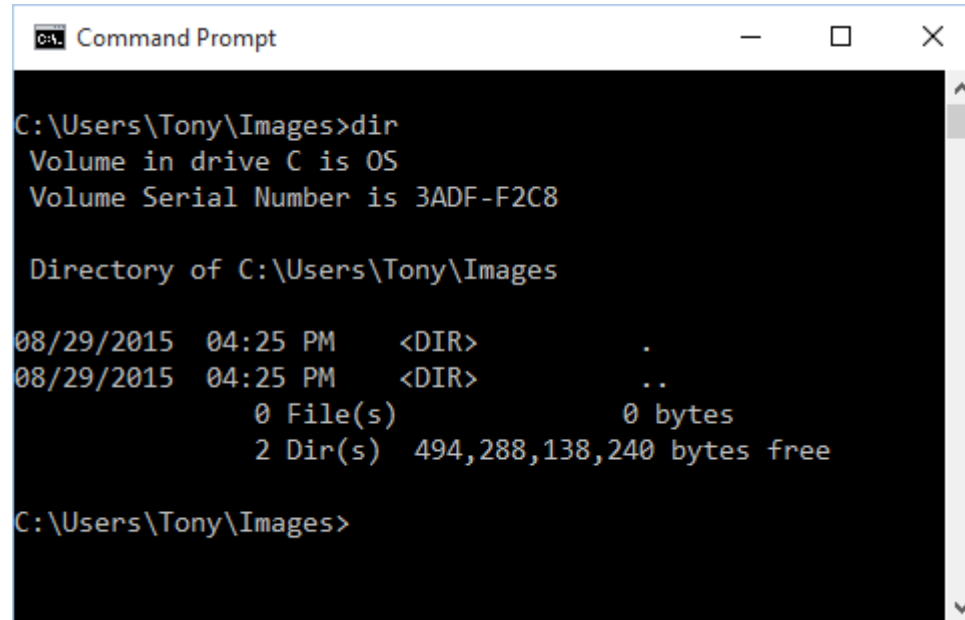
Una interfaz gráfica de usuario permite al usuario interactuar con el sistema operativo y otros programas utilizando elementos gráficos como iconos, botones y cuadros de diálogo.

Interfaces Gráficas de Usuario

- **Interfaz de Usuario** : la parte del computador con la cual interactúa el usuario.
- **Interfaz de línea de comando**: Muestra un prompt y el usuario tipea un comando que es luego ejecutado.
- **Interfaz Gráfica de Usuario(GUI)**: Permite a los usuarios interactuar con un programa a través de elementos graficos en la pantalla.

Interfaz de Linea de Comando

- Una interfaz de línea de **comando**.



```
C:\Users\Tony\Images>dir
Volume in drive C is OS
Volume Serial Number is 3ADF-F2C8

Directory of C:\Users\Tony\Images

08/29/2015  04:25 PM    <DIR>          .
08/29/2015  04:25 PM    <DIR>          ..
               0 File(s)                0 bytes
               2 Dir(s)  494,288,138,240 bytes free

C:\Users\Tony\Images>
```

Interfaces gráficas de usuario

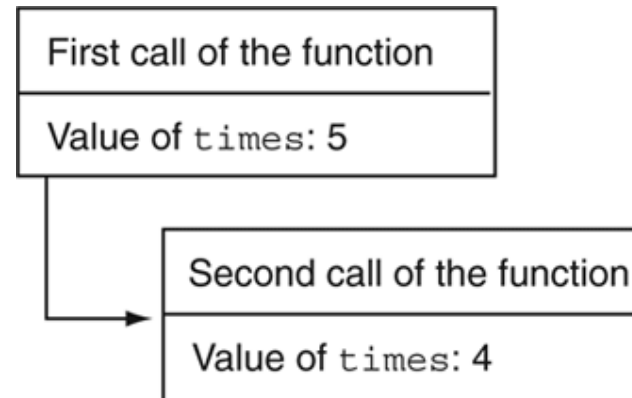
La interfaz de usuario de una computadora es la parte de la computadora con la que el usuario interactúa

La interfaz de usuario consta de

- Dispositivos de hardware
- Comandos del usuario que el sistema operativo acepta

Interfaz de línea de comando muestra un mensaje y el usuario escribe un comando que luego se ejecuta

Figura 07-1 Una Interfaz de línea de comando



Interfaces gráficas de usuario

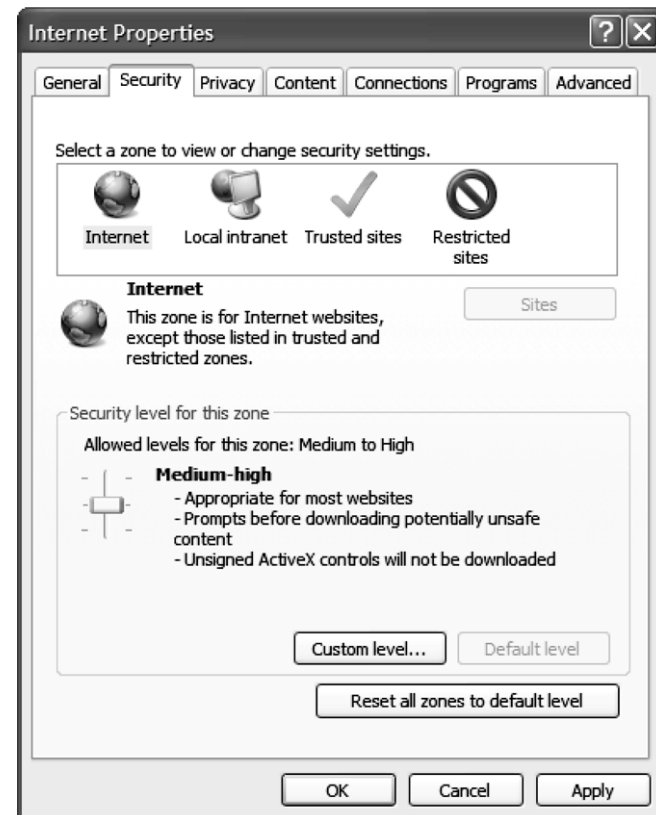
Una **interfaz gráfica del usuario (GUI)** permite al usuario interactuar con el sistema operativo y otros programas a través de elementos gráficos (iconos, botones, barras deslizantes, etc.) en la pantalla.

- GUIs popularizó el uso del mouse.
- GUIs permite al usuario apuntar a elementos gráficos y hacer clic en el botón del mouse para activarlos.

Interfaces gráficas de usuario

La interacción con una GUI se realiza a través de **cajas de diálogo** – pequeñas ventanas que muestran información y permiten al usuario realizar acciones

Figura 07-2 Una caja de diálogo



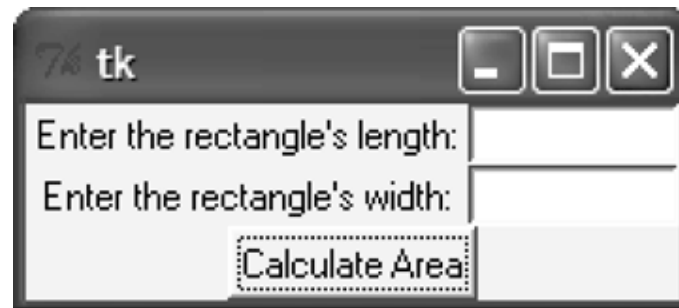
Interfaces gráficas de usuario

Los programas GUI están dirigidos por eventos.

El usuario determina el orden en que suceden las cosas.

Los programas GUI responden a las acciones del usuario, por lo tanto son dirigidos por eventos.

Figura 07-3 Un programa GUI



Usando el Módulo tkinter

Concepto:

En Python puede usar el módulo tkinter para crear programas GUI simples.

Usando el Módulo tkinter

- Python no tiene características de programación GUI integradas en el lenguaje.
- El módulo tkinter permite la creación de programas GUI simples.
- `tkinter` no siempre se ejecuta de manera confiable bajo IDLE
- Use el editor de IDLE para escribir programas GUI, pero para obtener mejores resultados, ejecute el programa desde la línea de comandos del sistema operativo.

widgets tkinter

Concepto:

tkinter proporciona varios controles, como botones, etiquetas y cuadros de texto utilizados en una aplicación GUI. Estos controles se denominan comúnmente **widgets**.

Actualmente hay 15 tipos de widgets en tkinter.

widgets tkinter

Concepto:

Los **widgets** son objetos que se pueden agregar a nuestra ventana de nivel superior. Esto permitirá al usuario interactuar con el programa. Algunos ejemplos de widgets:

Botones, botones de control, botones de radio, botones de menú,

Label (subtítulos de texto, imágenes)

Entry (para entradas de campo de texto)

Frame (un contenedor para otros widgets)

Scale, Scrollbar (barra de desplazamiento)

Canvas (para dibujar formas, ...)

Text (para mostrar y editar texto) y otros.

Table 13-1 `tkinter` Widgets

Widget	Description
Button	A button that can cause an action to occur when it is clicked.
Canvas	A rectangular area that can be used to display graphics.
Checkbutton	A button that may be in either the “on” or “off” position.
Entry	An area in which the user may type a single line of input from the keyboard.
Frame	A container that can hold other widgets.
Label	An area that displays one line of text or an image.
Listbox	A list from which the user may select an item
Menu	A list of menu choices that are displayed when the user clicks a <code>Menubutton</code> widget.
Menubutton	A menu that is displayed on the screen and may be clicked by the user
Message	Displays multiple lines of text.
Radiobutton	A widget that can be either selected or deselected. <code>Radiobutton</code> widgets usually appear in groups and allow the user to select one of several options.
Scale	A widget that allows the user to select a value by moving a slider along a track.
Scrollbar	Can be used with some other types of widgets to provide scrolling ability.
Text	A widget that allows the user to enter multiple lines of text input.
Toplevel	A container, like a <code>Frame</code> , but displayed in its own window.

Ventana de nivel superior: `tkinter.Tk()`

Todos los widgets principales están contruidos en el objeto ventana de nivel superior. Este objeto es creado por la clase **Tk** en `tkinter` y se instancia de la siguiente manera:

```
import tkinter  
ventana = tkinter.Tk()
```

Dentro de esta ventana, puede colocar widgets individuales o piezas de varios componentes para formar su GUI.

Usando el Módulo tkinter

Programa 07-1 (ventana_vacia.py)

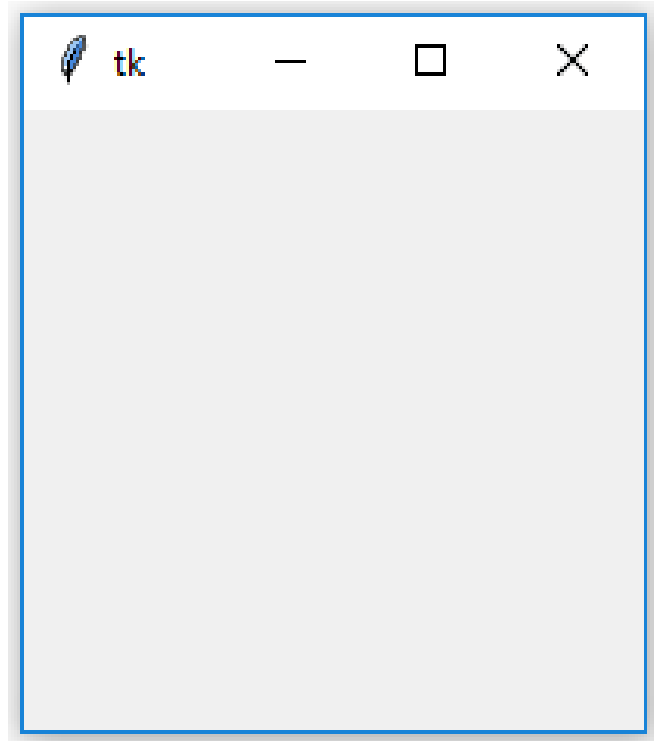
```
#Este programa muestra una ventana vacia
import tkinter

def main():
    #Crea el widget ventana
    ventana=tkinter.Tk()

    tkinter.mainloop()

#Llama a la función main
main()
```

Figura 07-4 Ventana mostrada por programa 07-1



Color Names, running, all screens

<https://wiki.tcl-lang.org/page/Color+Names%2C+running%2C+all+screens>

Color Names, running, all screens

Zipguy 2013-02-09 - You can find out my email address by clicking on [Zipguy](#).

This page has all of the screens from the program I copied from [Colors with Names](#).

Tk Named Colors				
snow	ghost white	white smoke	gainsboro	floral white
old lace	linen	antique white	papaya whip	blanched almond
bisque	peach puff	navajo white	moccasin	cornsilk
ivory	lemon chiffon	seashell	honeydew	mint cream
azure	alice blue	lavender	lavender blush	misty rose
white	black	dark slate gray	dim gray	slate gray
light slate gray	gray	light gray	midnight blue	navy
cornflower blue	dark slate blue	slate blue	medium slate blue	light slate blue
medium blue	royal blue	blue	dodger blue	deep sky blue
sky blue	light sky blue	steel blue	light steel blue	light blue
powder blue	pale turquoise	dark turquoise	medium turquoise	turquoise
cyan	light cyan	cadet blue	medium aquamarine	aquamarine
dark green	dark olive green	dark sea green	sea green	medium sea green
light sea green	pale green	spring green	lawn green	green
chartreuse	medium spring green	green yellow	lime green	yellow green
forest green	olive drab	dark khaki	khaki	pale goldenrod
light goldenrod yellow	light yellow	yellow	gold	light goldenrod
goldenrod	dark goldenrod	rosy brown	indian red	saddle brown
sienna	peru	burlywood	beige	wheat
sandy brown	tan	chocolate	firebrick	brown
dark salmon	salmon	light salmon	orange	dark orange
coral	light coral	tomato	orange red	red
hot pink	deep pink	pink	light pink	pale violet red
maroon	medium violet red	violet red	magenta	violet
plum	orchid	medium orchid	dark orchid	dark violet
blue violet	purple	medium purple	thistle	snow2
snow3	snow4	seashell2	seashell3	seashell4
AntiqueWhite1	AntiqueWhite2	AntiqueWhite3	AntiqueWhite4	bisque2
bisque3	bisque4	PeachPuff2	PeachPuff3	PeachPuff4
NavajoWhite2	NavajoWhite3	NavajoWhite4	LemonChiffon2	LemonChiffon3

lavender => 59110 / 59110 / 64250 => #e6e6fa

El método mainloop

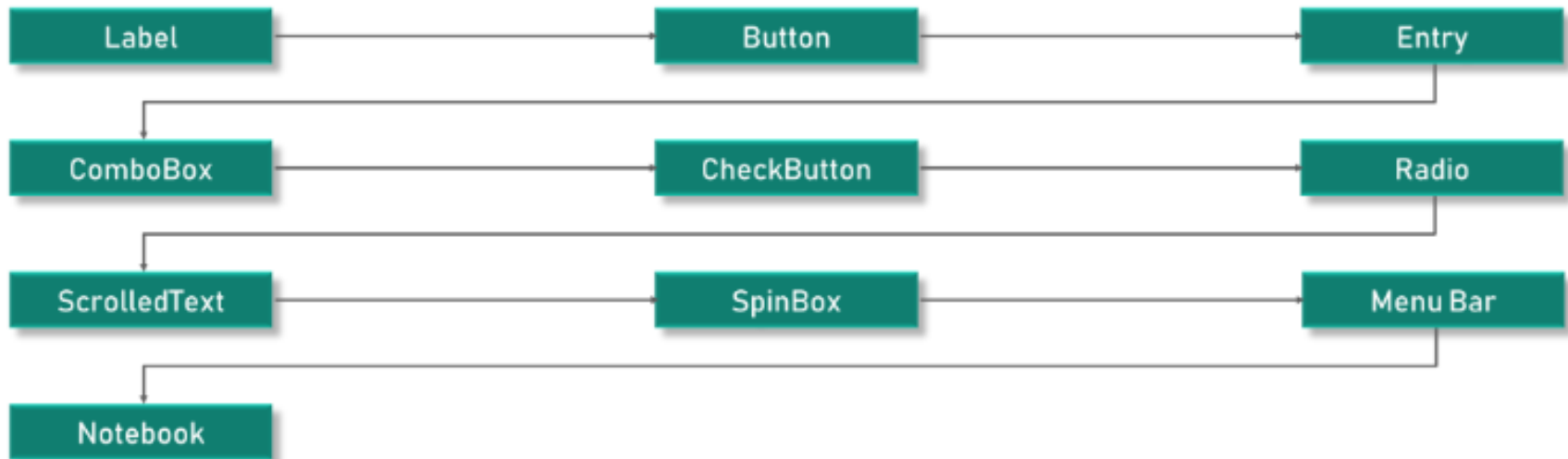
Concepto:

El método **mainloop ()** inicia el bucle principal, que rastrea el mouse y determina cuándo y dónde se presionó. **mainloop ()** es un bucle infinito que se usa para ejecutar la aplicación, espera a que ocurra un evento y procesa el evento mientras la ventana no se cierre.

mainloop ():

- Existe un método conocido por el nombre `mainloop ()` que se usa cuando estamos listos para ejecutar la aplicación.
- `mainloop ()` es un bucle infinito que se usa para ejecutar la aplicación.
- Espera a que ocurra un evento y procesa el evento hasta que la ventana no se cierre.

Principales widgets tkinter



Lista de opciones Ventana de Nivel Superior

background: establece el color de fondo normal.

geometry: establece la dimension de la ventana.

title: establece el titulo de la ventana.

Mostrar texto con widgets Label

Concepto:

Utilice el widget **Label** para mostrar texto en una ventana.

Mostrar texto con widgets Label

- Este widget implementa un cuadro de visualización donde puede colocar texto o imágenes. El texto que muestra este widget se puede actualizar en cualquier momento que desee.

Sintaxis:

Aquí está la sintaxis simple para crear este widget:

```
w = Label ( master, opciones, ... )
```

Parámetros:

- **master:** Representa la ventana principal.
- **opciones:** Lista de las opciones más utilizadas para este widget. Estas opciones se pueden usar como pares clave-valor separados por comas.

Lista de opciones Widget Label

bg: establecer el color de fondo normal.

fg: establecer el color de texto.

font: establecer la fuente en la etiqueta.

text: Especifica una cadena de texto que se mostrará dentro del widget.

textvariable: Especifica un nombre cuyo valor se utilizará en lugar del recurso de opción text.

Posicionamiento de widgets

El empaquetador es uno de los mecanismos de gestión de geometría de Tk. Los **gestores de geometría** se utilizan para especificar el posicionamiento de widgets dentro de su contenedor. El empaquetador toma la especificación de relación cualitativa, arriba, a la izquierda, relleno, etc., y resuelve todo para determinar las coordenadas exactas de ubicación.

Tenga en cuenta que **los widgets no aparecen** hasta que se haya especificado su geometría con un **gestor de geometría**. Es un error común dejar de lado el especificador de geometría y luego sorprenderse cuando se crea el widget pero no aparece nada. Un widget aparecerá solo después de que se le haya aplicado el empaquetador, por ejemplo, el método **pack ()**

El método pack()

Se puede llamar al método **pack ()** con pares **clave - valor** que controlan dónde debe aparecer el widget dentro de su contenedor y cómo se comportará cuando se cambie el tamaño de la ventana principal de la aplicación. A continuación algunos ejemplos:

```
label.pack()           #predeterminado side =  
"TOP"
```

```
label.pack(side = "LEFT")
```

```
label.pack(expand = 1)
```

Lista de opciones del Método pack

anchor: Tipo de anclaje. Denota dónde el empaquetador colocará cada esclavo en su parcela.

expand: Booleano, 0 o 1.

fill: Valores legales: 'x', 'y', 'BOTH', 'NONE'.

ipadx, ipady: Una distancia que designa el relleno interno a cada lado del widget esclavo.

padx, pady: Una distancia que designa el relleno externo a cada lado del widget esclavo.

side: Los valores legales son: 'LEFT', 'RIGHT', 'TOP', 'BOTTOM'.

Método Tkinter pack ()

pack() - Posicionamiento relativo

- Este gestor de geometría organiza los widgets en bloques antes de colocarlos en el widget principal.
- Le decimos que el elemento debe ir arriba, abajo, a la izquierda o a la derecha respecto de algún otro control o bien la ventana principal.

pack(side)

```
import tkinter
windows=tkinter.Tk()
windows.title("Mi primera GUI")
windows.geometry("700x400")
windows.configure(background="IndianRed4")
label=tkinter.Label(windows,text="Welcome to Python")
label.pack(side="right")
```

Ejemplo: agregar un widgets Label

- **Crea una etiqueta**

Al crear el widget **Label**, debemos pasar la ventana de nivel superior, **ventana**, (en la que se colocará la etiqueta) como primer argumento.

También debemos pasar el texto que se mostrará dentro de la etiqueta como segundo argumento.

- **Defina la posición de la etiqueta (`self.label.pack ()`)**

Indique a la etiqueta que se coloque en la ventana raíz y que se muestre.

- **Inicie el bucle de eventos (`tkinter.mainloop ()`)**

Mostrar texto con widgets Label

Programa 07-3 (hola_mundo.py)

```
#Este programa muestra una etiqueta con texto
import tkinter
class MiGUI:
    def __init__(self):
        #Crea el widget ventana
        self.ventana=tkinter.Tk()
        #Crea el widget label
        self.label=tkinter.Label(self.ventana,text="Hola Mundo!")
        #Llama al metodo pack del widget Label
        self.label.pack()
        tkinter.mainloop()

#Crea una instancia de la clase MiGUI
mi_gui=MiGUI()
```

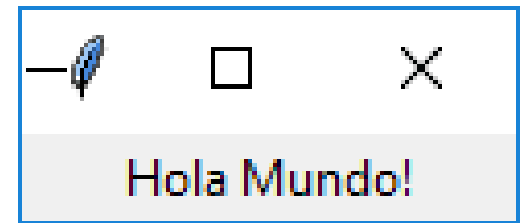


Figura 07-5 Ventana mostrada por el Programa 07-3

Mostrar texto con widgets Label

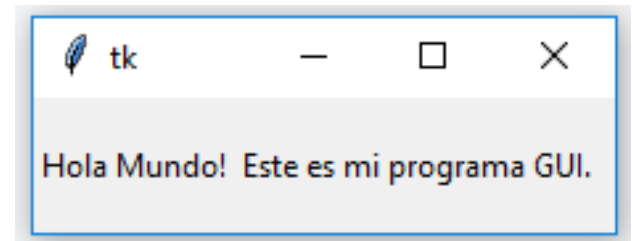
Programa 07-4 (hola_mundo2.py)

Figura 07-7 Ventana mostrada por Programa 07-4

```
#Este programa usa el argumento side='left' con
#el metodo pack para cambiar el diseño de los widgets
import tkinter
class MiGUI:
    def __init__(self):
        #Crea el widget ventana
        self.ventana=tkinter.Tk()
        #Crea dos widget label
        self.label1=tkinter.Label(self.ventana,text="Hola Mundo!")
        self.label2=tkinter.Label(self.ventana,text="Este es mi programa GUI.")
        #Llama al metodo pack para cada widget Label
        self.label1.pack(side='left')
        self.label2.pack(side='left')

        tkinter.mainloop()

#Crea una instancia de la clase MiGUI
mi_gui=MiGUI()
```



Widgets Button

Concepto:

Use el widget `Button` para crear un botón estándar en una ventana. Cuando el usuario hace clic en un botón, se llama a una función o método específico.

Sintaxis Widget Button

El widget Button se usa para agregar botones en una aplicación Python. Puede adjuntar una función o un método a un botón, que se llama automáticamente cuando hace clic en el botón.

Sintaxis:

```
w = Button ( master, opcion=value, ... )
```

Parámetros:

- **master:** representa la ventana principal.
- **opciones:** lista de las opciones más utilizadas para este widget. Estas opciones se pueden usar como pares clave-valor separados por comas.

Lista de opciones Widget Button

activebackground: establecer el color de fondo cuando el botón está debajo del cursor.

activeforeground: establecer el color de primer plano cuando el botón está debajo del cursor.

bg: establecer el color de fondo normal.

command: llamar a una función.

font: establecer la fuente en la etiqueta del botón.

image: establecer la imagen en el botón.

width: establecer el ancho del botón.

height: establecer la altura del botón.

Widgets Button

- Un Button es un widget en el que el usuario puede hacer clic para provocar una acción.
- Una *función de devolución de llamada (controlador de eventos)* es una función o método que se ejecuta cuando el usuario hace clic en el botón.

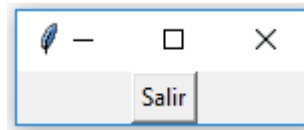
Insertando el widget Button

Programa 07-5 (boton_salir.py)

```
import tkinter
class MiGUI:
    def __init__(self):
        ventana=tkinter.Tk()
        self.boton=tkinter.Button(ventana,text="Salir", command=ventana.destroy)
        self.boton.pack()
        tkinter.mainloop()

#Crea el widget ventana
mi_gui=MiGUI()
```

Figura 07-8 Ventana mostrada por programa 07-5



Método **destroy**

El método **destroy** se puede llamar en cualquier widget de Tkinter y lo eliminará de la aplicación.

Destruir la ventana principal hará que la aplicación se cierre, así que se debe usar con cuidado.

Método **say_goodbye**

- El método **say_goodbye** actualizará el texto de la etiqueta y luego cerrará la ventana después de dos segundos.
- Esto lo logramos usando el método **after** de nuestro widget Tk (que hemos subclasificado en Window).
- Este método llamará a un fragmento de código una vez transcurrido un cierto tiempo (en milisegundos).

Método after

after (demora, callback = None)

Es un método definido para todos los widgets tkinter. Este método simplemente llama a la función de **devolución de llamada** después de la **demora** dada en ms. Si no se proporciona ninguna función, actúa de forma similar a time.sleep (pero en milisegundos en lugar de segundos)

Programación GUI

PREGUNTAS

?