Interfaz Gráfica de Usuario

PROGRAMACIÓN EN

Python

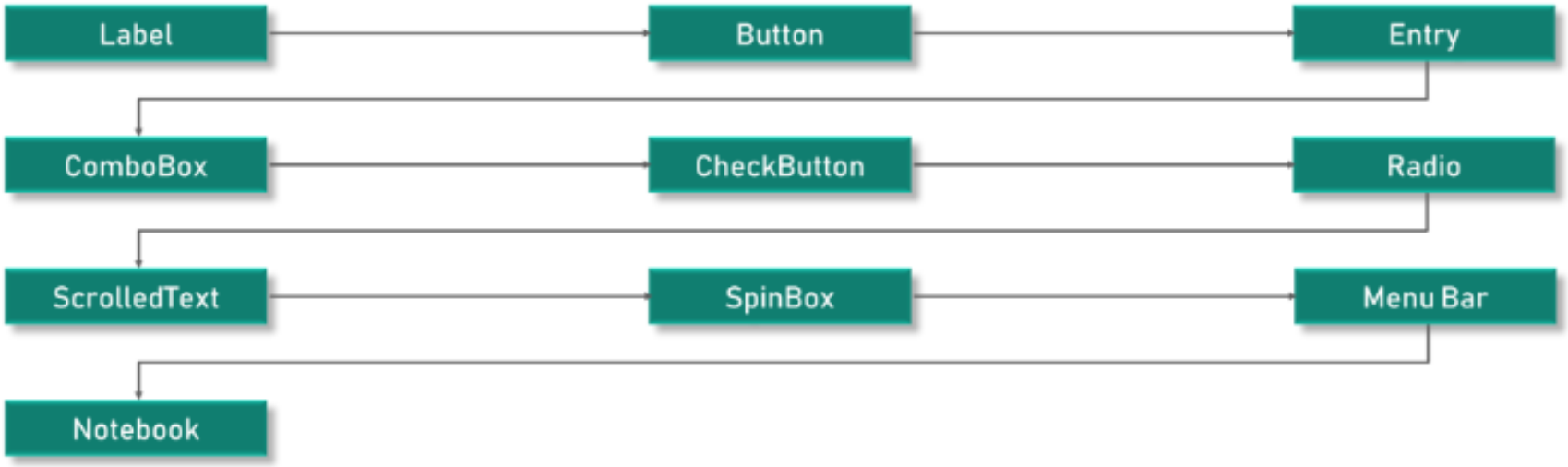
por
Víctor Melchor

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.

Principales widgets tkinter



Organizar widgets con Frames

Concepto:

Un **`Frame`** es un contenedor que puede contener otros widgets. Puede usar Frames para organizar los widgets en una ventana.

Organizar widgets con Frames

- Un `Frame` es un contenedor.
- Es un widget que puede contener otros widgets.
- Los `Frames` son útiles para organizar y clasificar grupos de widgets en una ventana.

Organizar widgets con Frames

Programa 07-6 (frame_demo.py)

```
#Este programa crea labels en dos marcos diferentes
import tkinter
class MiGUI:
    def __init__(self):
        #Crea el widget ventana
        self.ventana=tkinter.Tk()
        #Crea dos frames, uno para la parte superior de la ventana
        #y otro para la parte inferior
        self.frameSup=tkinter.Frame(self.ventana)
        self.frameInf=tkinter.Frame(self.ventana)

        #Crea tres widgets Label para el frame Superior
        self.label1=tkinter.Label(self.frameSup,text="Uno")
        self.label2=tkinter.Label(self.frameSup,text="Dos")
        self.label3=tkinter.Label(self.frameSup,text="Tres")
        #Llama al metodo pack para cada widget Label
        self.label1.pack(side='top')
        self.label2.pack(side='top')
        self.label3.pack(side='top')
```

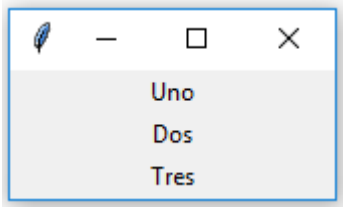


Figura 07-8 Ventana mostrada por Programa 07-6

Organizar widgets con Frames

Programa 07-6 (frame_demo.py)

```
        #Crea tres widgets Label para el frame Inferior
        self.label4=tkinter.Label(self.frameInf,text="Uno")
        self.label5=tkinter.Label(self.frameInf,text="Dos")
        self.label6=tkinter.Label(self.frameInf,text="Tres")
        #Llama al metodo pack para cada widget Label
        self.label4.pack(side='left')
        self.label5.pack(side='left')
        self.label6.pack(side='left')

        #Llama al metodo pack para cada Frame tambien
        self.frameSup.pack()
        self.frameInf.pack()

        tkinter.mainloop()

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

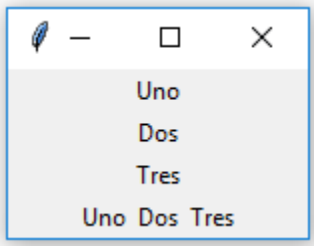


Figura 07-9 Ventana mostrada por Programa 07-6

Organizar widgets con Frames

Figura 07-10 Ventana mostrada por el programa 07-6

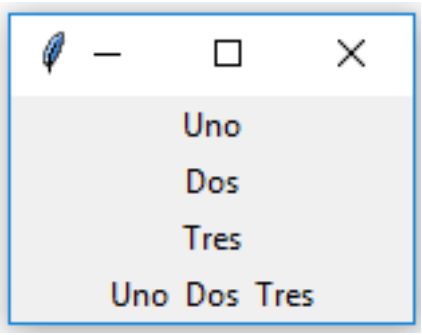
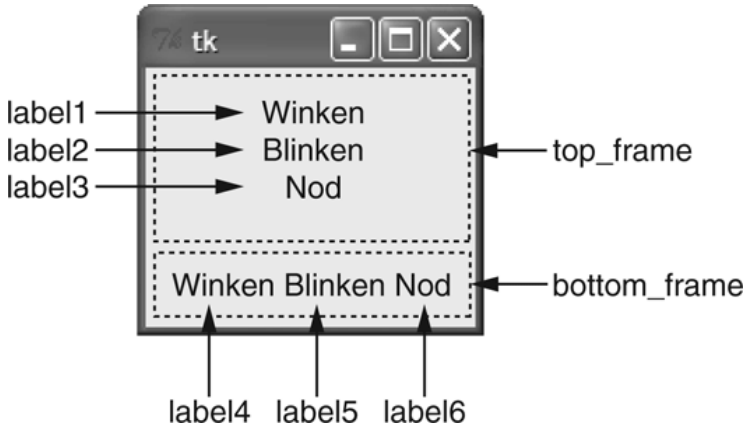


Figura 07-11 Disposición de widgets



Obteniendo entrada con el widget `Entry`

Concepto:

Un widget **Entry** es un área rectangular en la que el usuario puede escribir la entrada. Use el método get del widget Entry para recuperar los datos que se han escrito en el widget.

Sintaxis del widget `Entry`

Python - Tkinter Entry:

- El widget Entry se usa para aceptar cadenas de texto (que ocupan una sola línea) de un usuario.
- Si desea mostrar varias líneas de texto que se pueden editar, debe usar el widget Text.
- Si desea mostrar una o más líneas de texto que el usuario no puede modificar, entonces debe usar el widget Label.

Syntaxis:

Aquí está la sintaxis simple para crear este widget:

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

Parámetros:

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

Opciones para configurar el widget `Entry`

- bd: establecer el ancho del borde en píxeles.
- bg: establecer el color de fondo normal.
- cursor: configurar el cursor utilizado.
- command: llamar a una función.
- highlightcolor: establecer el color que se muestra en el resaltado de enfoque.
- width: establecer el ancho del botón.
- height: establecer la altura del botón.

Obteniendo entrada con el widget `Entry`

```
#Uso de widgets Label y Entry
import tkinter
class Mi_GUI():
    def __init__(self):
        self.window=tkinter.Tk()
        self.frame_sup=tkinter.Frame(self.window)
        self.frame_inf=tkinter.Frame(self.window)

        self.lblmsj=tkinter.Label(self.frame_sup,\
                                  text="Ingrese una distancia en Km.= ")
        self.entry=tkinter.Entry(self.frame_sup,width=10)
        self.lblmsj.pack(side='left')
        self.entry.pack(side='left')

        self.btn_conv=tkinter.Button(self.frame_inf,text='Convertir',command=self.convertir)
        self.btn_salir=tkinter.Button(self.frame_inf,text='Salir',command=self.window.destroy)
```

Programa 12-9
conversor_kilo.py)

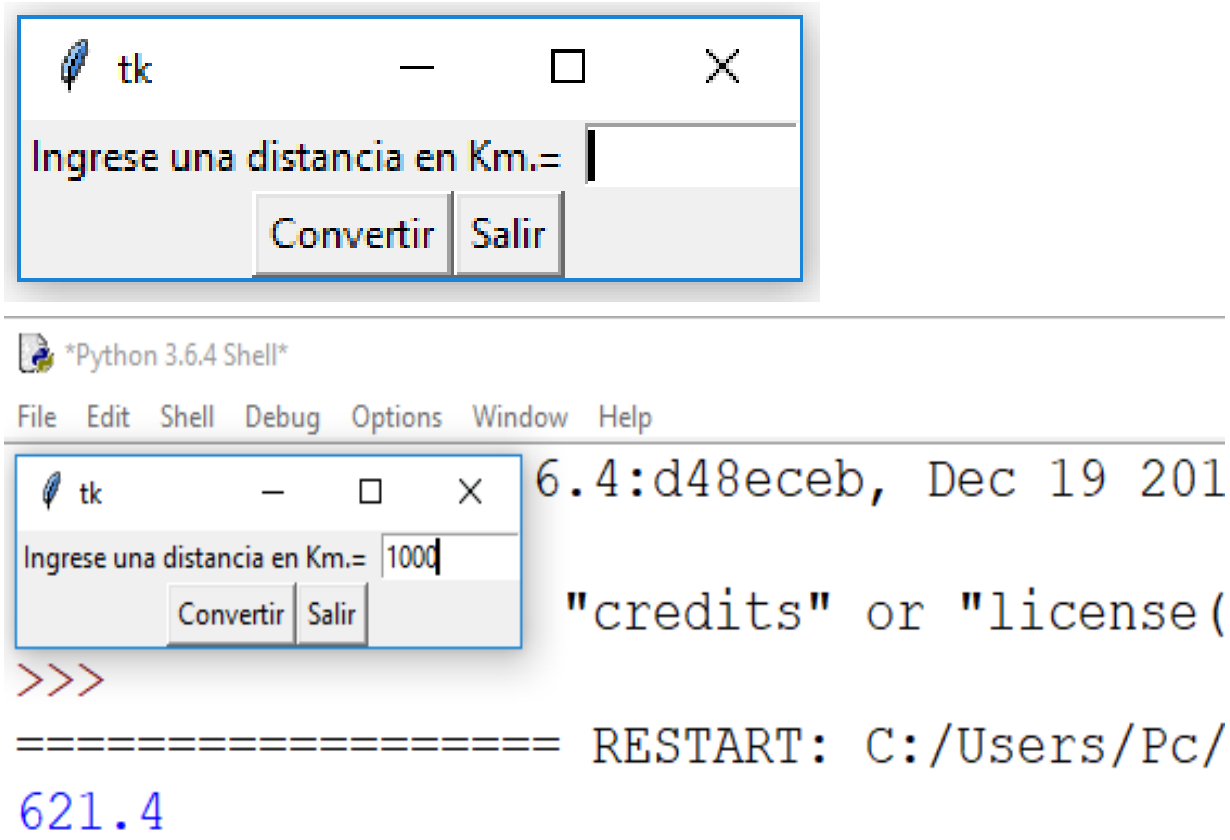
Obteniendo entrada con el widget `Entry`

```
        self.btn_conv=tkinter.Button(self.frame_inf,text='Convertir',command=self.convertir)
        self.btn_salir=tkinter.Button(self.frame_inf,text='Salir',command=self.window.destroy)

        #Posicionando los botones
        self.btn_conv.pack(side='left')
        self.btn_salir.pack(side='left')

        self.frame_sup.pack()
        self.frame_inf.pack()
        #Inicia el mainloop
        tkinter.mainloop()

#     El método convertir es una funcion callback para el boton convertir
    def convertir(self):
        kilo=float(self.entry.get())
        millas=kilo*0.6214
        #Convierte kilo a millas y lo muestra en el interprete
        print(millas)

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

Programa 12-9
conversor_kilo.py)

Conversión de km a millas

Figura 12-15 Envío de la conversión de Km. al intérprete



Usar Variables de Control

Concepto:

Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valores y facilitar su disponibilidad en otras partes del programa. Pueden ser de tipo numérico, de cadena y booleano.

Cuando una variable de control cambia de valor el widget que la utiliza lo refleja automáticamente, y viceversa.

Declarar Variables de Control

Las variables de control se declaran de forma diferente en función al tipo de dato que almacenan:

```
entero = IntVar()          # Declara variable de tipo entera
flotante = DoubleVar()     # Declara variable de tipo flotante
cadena = StringVar()       # Declara variable de tipo cadena
booleano = BooleanVar()  # Declara variable booleana
```

Usar Labels como campos de salida

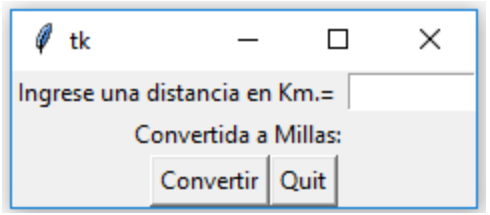
Concepto:

Cuando un objeto `StringVar` está asociado con un widget `Label`, el widget `Label` muestra cualquier dato que es almacenado en el objeto `StringVar`.

Usar Labels como campos de salida

Programa 12-10 (convertir_kilos.py)

```
#Uso de widgets Label y Entry
import tkinter
class Mi_GUI():
    def __init__(self):
        self.window=tkinter.Tk()
        self.frame_sup=tkinter.Frame(self.window)
        self.frame_medio=tkinter.Frame(self.window)
        self.frame_inf=tkinter.Frame(self.window)

        self.lbldist=tkinter.Label(self.frame_sup,text="Ingrese una distancia en Km.= ")
        self.entry=tkinter.Entry(self.frame_sup,width=10)
        self.lbldist.pack(side='left')
        self.entry.pack(side='left')

        self.lbldescr=tkinter.Label(self.frame_medio,text="Convertida a Millas:")
        self.valor=tkinter.StringVar()
        self.lblmillas=tkinter.Label(self.frame_medio,textvariable=self.valor)
        #Posicionando las etiquetas
        self.lbldescr.pack(side='left')
        self.lblmillas.pack(side='left')
```

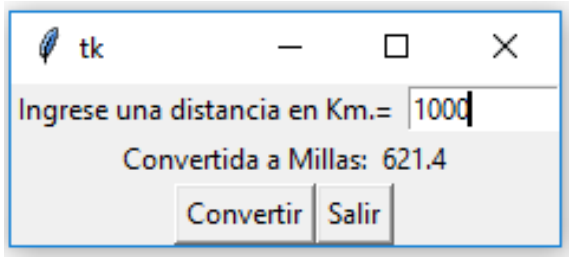


Usar Labels como campos de salida

Programa 12-10 (convertir_kilos.py)

```
        #Posicionando las etiquetas
        self.lbldescr.pack(side='left')
        self.lblmillas.pack(side='left')

        self.btn_conv=tkinter.Button(self.frame_inf,text='Convertir',command=self.convertir)
        self.btn_salir=tkinter.Button(self.frame_inf,text='Salir',command=self.window.destroy)

        #Posicionando los botones
        self.btn_conv.pack(side='left')
        self.btn_salir.pack(side='left')

        self.frame_sup.pack()
        self.frame_medio.pack()
        self.frame_inf.pack()
        #Inicia el mainloop
        tkinter.mainloop()
    def convertir(self):
        kilo=float(self.entry.get())
        millas=kilo*0.6214
        #Convierte kilo a millas y lo asigna en el objeto StringVar
        #Este automaticamente actualizara el widget lblmillas
        self.valor.set(millas)

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

Figura 12-18
Diseño de la ventana principal del programa `convertir_kilos`



Radio Buttons y Check Buttons

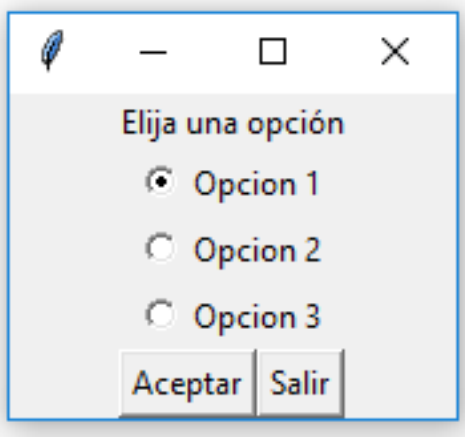
Concepto:

Los botones de opción(Radio Buttons) normalmente aparecen en grupos de dos o más y permiten al usuario seleccionar una de varias opciones posibles. Los botones de verificación(Check Buttons), que pueden aparecer solos o en grupos, permiten al usuario hacer selecciones del tipo sí / no o activar / desactivar.

Radio Buttons y Check Buttons

Radio buttons **se usan cuando el usuario necesita seleccionar una opción de varias opciones.**

Figura 12-22 Un grupo de radio buttons



Radio Buttons y Check Buttons

Programa 12-12
(radiobutton_demo.py)

```
#Uso de botones de radio
import tkinter

class Mi_GUI():
    def __init__(self):
        self.window=tkinter.Tk()
        self.top_frame=tkinter.Frame(self.window)
        self.bottom_frame=tkinter.Frame(self.window)

        self.radio_var=tkinter.IntVar()
        self.radio_var.set(1)
        self.rb1=tkinter.Radiobutton(self.top_frame,text="Opcion 1",variable=self.radio_var,value=1)
        self.rb2=tkinter.Radiobutton(self.top_frame,text="Opcion 2",variable=self.radio_var,value=2)
        self.rb3=tkinter.Radiobutton(self.top_frame,text="Opcion 3",variable=self.radio_var,value=3)

        self.btnOK=tkinter.Button(self.bottom_frame,text='Aceptar',command=self.show_choice)
        self.btnSalir=tkinter.Button(self.bottom_frame,text='Salir',command=self.window.destroy)
        #Posicionando los botones
        self.btnOK.pack(side='left')
        self.btnSalir.pack(side='left')
```

Radio Buttons y Check Buttons

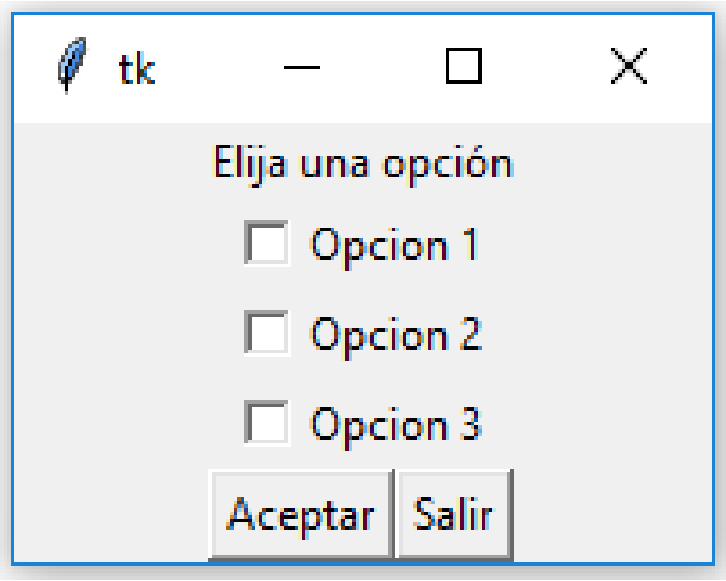
Programa 12-12
(radiobutton_demo.py)

```
        #Posicionando los botones
        self.btnOK.pack(side='left')
        self.btnSalir.pack(side='left')

        #Posicionando los radio botones
        self.rb1.pack()
        self.rb2.pack()
        self.rb3.pack()

        self.lblMsj=tkinter.Label(self.window,text="Elija una opción")
        #Posicionando la etiqueta
        self.lblMsj.pack()

        self.top_frame.pack()
        self.bottom_frame.pack()
        #Inicia el mainloop
        tkinter.mainloop()
    def show_choice(self):
        self.lblMsj.configure(text="Elegiste la opcion "+str( self.radio_var.get() ) )

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

Radio Buttons y Check Buttons

Check buttons **se usan cuando el usuario puede hacer múltiples elecciones.**

Figura 12-24 Un grupo de check buttons



Uso de Check Buttons

Programa 12-13
(checkbutton_demo.py)

```
#Uso de botones de Verificacion
import tkinter

class Mi_GUI():
    def __init__(self):
        self.window=tkinter.Tk()
        self.frame_sup=tkinter.Frame(self.window)
        self.frame_inf=tkinter.Frame(self.window)
        self.cb_var1=tkinter.IntVar()
        self.cb_var2=tkinter.IntVar()
        self.cb_var3=tkinter.IntVar()

        self.cb_var1.set(0)
        self.cb_var2.set(0)
        self.cb_var3.set(0)

        self.cb1=tkinter.Checkbutton(self.frame_sup,text="Opcion 1",variable=self.cb_var1)
        self.cb2=tkinter.Checkbutton(self.frame_sup,text="Opcion 2",variable=self.cb_var2)
        self.cb3=tkinter.Checkbutton(self.frame_sup,text="Opcion 3",variable=self.cb_var3)

        #Posicionando los Check botones
        self.cb1.pack()
        self.cb2.pack()
        self.cb3.pack()
```

Radio Buttons y Check Buttons

```
        #Posicionando los Check botones
        self.cb1.pack()
        self.cb2.pack()
        self.cb3.pack()

        self.btnOK=tkinter.Button(self.frame_inf,text='Aceptar',command=self.muestra_eleccion)
        self.btnSalir=tkinter.Button(self.frame_inf,text='Salir',command=self.window.destroy)
        #Posicionando los botones
        self.btnOK.pack(side='left')
        self.btnSalir.pack(side='left')

        self.lblMsj=tkinter.Label(self.window,text="Elija una opción")
        #Posicionando la etiqueta
        self.lblMsj.pack()

        self.frame_sup.pack()
        self.frame_inf.pack()
        #Inicia el mainloop
        tkinter.mainloop()
    def muestra_eleccion(self):
        self.mensaje="Elegiste\n"
        if self.cb_var1.get()==1:
            self.mensaje=self.mensaje+" 1\n"
        if self.cb_var2.get()==1:
            self.mensaje=self.mensaje+" 2\n"
        if self.cb_var3.get()==1:
            self.mensaje=self.mensaje+" 3\n"
        self.lblMsj.configure(text=self.mensaje)

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

Programa 12-13
(checkbutton_demo.py)

PREGUNTAS

?