Introducción a Tkinter en Python

Desarrollando Interfaces Gráficas de Usuario

Programación Avanzada - Prof. Felipe Morales

Avalos Maia - Lucca Pérez Veltri - Euler Diego

unab

Temario :

1. Tkinter: Introducción
2. Características y Ventajas de Tkinter
3. Características Avanzadas de Tkinter
4. Estructura Básica de una GUI en Tkinter
5. ¿Qué es un Widget en Tkinter?
6. Widgets Básicos en Tkinter
7. Posicionamiento de Widgets
8. Modificación de Widgets
9. Uso de Messagebox
10. Trabajando con Imágenes
11. Desarrollo de Aplicaciones

1. Tkinter: Introducción

Definición de Tkinter:

- Tkinter es un marco de interfaz gráfica integrado en la biblioteca estándar de Python.
- Actúa como puente entre Python y la librería TCL/TK.
- Es multiplataforma: los elementos se representan utilizando componentes nativos de cada sistema operativo.

Verificación de la Instalación:

- Podemos verificar la instalación de Tkinter ejecutando la siguiente línea en la consola:

```
python -m tkinter
```

2. Características y Ventajas de Tkinter:

Características Clave:

- **Multiplataforma:** Funciona en diferentes sistemas operativos utilizando componentes nativos.
- **Facilidad de Uso:** Comparado con otros marcos, es fácil de aprender y usar.
- **Ligero:** Es un marco liviano, ideal para desarrollar aplicaciones GUI rápidas y funcionales.
- **Biblioteca Estándar:** No requiere instalaciones adicionales, ya que está incluida en Python

Ventajas de Usar Tkinter:

- **Documentación y Comunidad:** Amplia documentación y una comunidad activa para soporte.

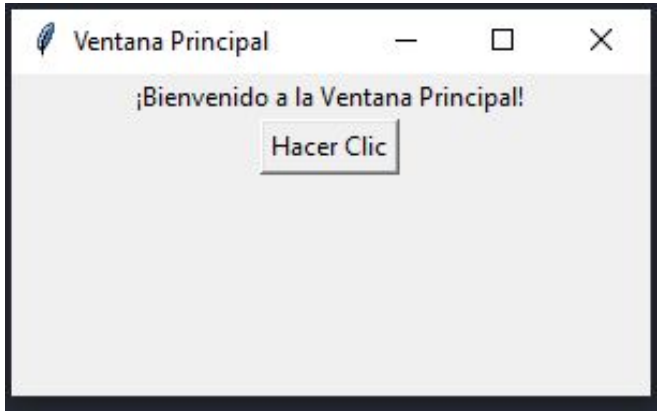
3. Características Avanzadas de Tkinter:

- **Widget Personalizables:** Gran variedad de widgets (botones, etiquetas, cuadros de texto, etc.) personalizables.
- **Soporte para Eventos:** Manejo de eventos como clics de ratón, teclas presionadas, etc.
- **Diseño de Layout Flexible:** Uso de gestores de geometría como pack, grid y place para organizar los widgets.
- **Extensibilidad:** Posibilidad de crear widgets personalizados y extender funcionalidades.
- **Soporte para Temas:** Permite cambiar el aspecto de la interfaz utilizando diferentes temas.

4. Estructura Básica de una GUI en Tkinter

4.1 La Ventana

Es el elemento fundamental de una GUI.



Contenedor Principal: Todas las interacciones y componentes de la GUI se agrupan dentro de la ventana.

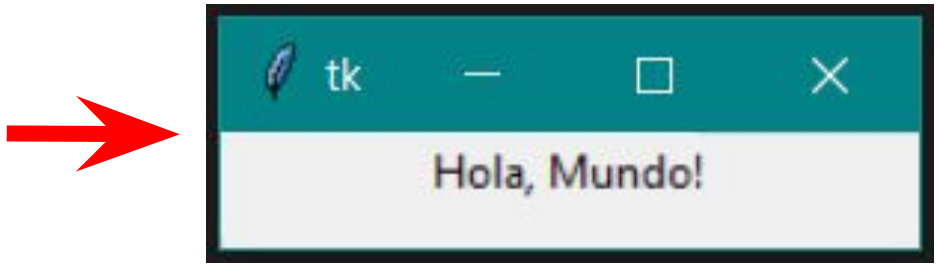
Widgets: Los elementos dentro de la ventana, como cuadros de texto, etiquetas y botones, son denominados widgets.

Gestión de Elementos: Permite organizar y gestionar la disposición de los widgets mediante diferentes gestores de geometría (pack, grid, place).

4.2 Pasos Básicos para Crear una Aplicación

1. Importar Tkinter
2. Crear una Ventana Principal
3. Añadir Widgets a la Ventana
4. Iniciar el Bucle de Eventos

```
import tkinter as tk

root = tk.Tk()
label = tk.Label(root, text="Hola, Mundo!")
label.pack()
root.mainloop()
```



5. ¿Qué es un Widget en Tkinter?

Es un componente de la interfaz gráfica de usuario (GUI) que permite la interacción entre el usuario y la aplicación.

Características Clave de los Widgets:

1. **Interactividad:**
   - Permiten la interacción del usuario con la aplicación a través de acciones como clics, entradas de texto, selecciones, etc.

2. **Visualización:**
   - Muestran información al usuario de manera clara y organizada.

3. **Organización:**
   - Ayudan a organizar otros widgets dentro de la ventana.

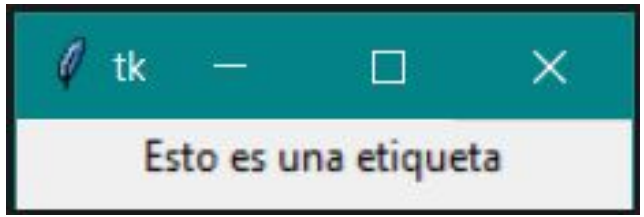
4. **Control de Eventos:**
   - Pueden responder a eventos específicos como clics de ratón, presiones de teclas, etc.
   - Permiten programar comportamientos dinámicos en la aplicación.

6. Widgets Básicos en Tkinter

6.1 Label

Muestra texto o una imagen, pero no es interactivo.

```
import tkinter as tk

ventana = tk.Tk()
# Crear una etiqueta con el texto "Esto es una etiqueta"
etiqueta = tk.Label(ventana, text="Esto es una etiqueta")
etiqueta.pack()  # Colocar la etiqueta en la ventana
ventana.mainloop()
```



6.2 Frame

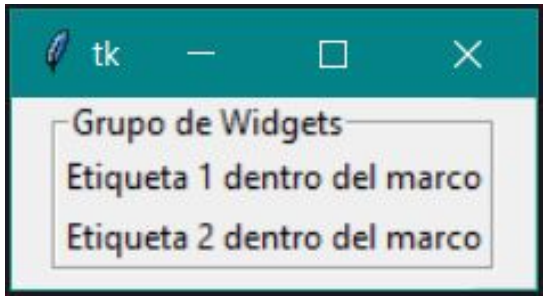
Es un contenedor rectangular invisible o visible (dependiendo de cómo lo implementamos), que puede contener otros widgets y ayudar a organizarlos.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un marco definiendo el grosor y estilo del borde
marco = tk.Frame(ventana, bd=1, relief="solid")
marco.pack()  # Colocar el marco en la ventana
# Crear una etiqueta dentro del marco
etiqueta = tk.Label(marco, text="Dentro del marco")
etiqueta.pack()  # Colocar la etiqueta en el marco
ventana.mainloop()
```

6.3 LabelFrame

Es un marco que contiene una etiqueta y puede contener otros widgets agrupados.

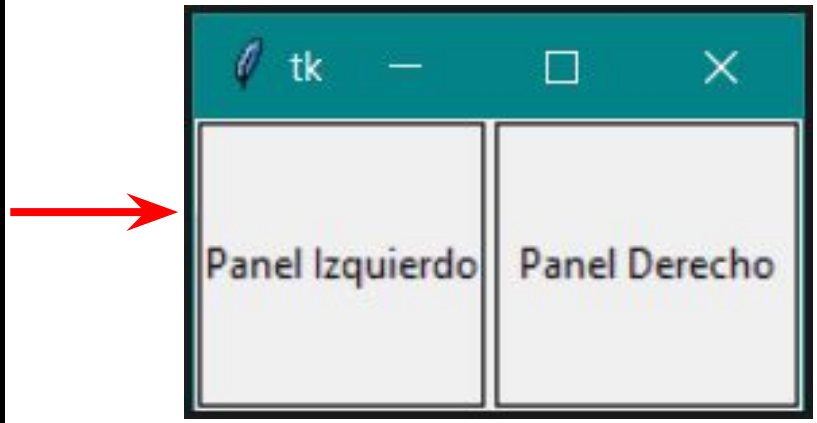
```
import tkinter as tk

ventana = tk.Tk()
# Crear un marco con etiqueta "Grupo de Widgets"
marco_etiqueta = tk.LabelFrame(ventana, text="Grupo de Widgets")
marco_etiqueta.pack() # Colocar el marco con etiqueta en la ventana
# Crear una etiqueta dentro del marco con etiqueta
etiqueta1 = tk.Label(marco_etiqueta, text="Etiqueta 1 dentro del marco")
etiqueta1.pack() # Colocar la primera etiqueta en el marco con etiqueta
etiqueta2 = tk.Label(marco_etiqueta, text="Etiqueta 2 dentro del marco")
etiqueta2.pack() # Colocar la segunda etiqueta en el marco con etiqueta
ventana.mainloop()
```



6.4 PanelWindow

Es un widget que contiene dos o más sub-paneles ajustables, separados por una barra.

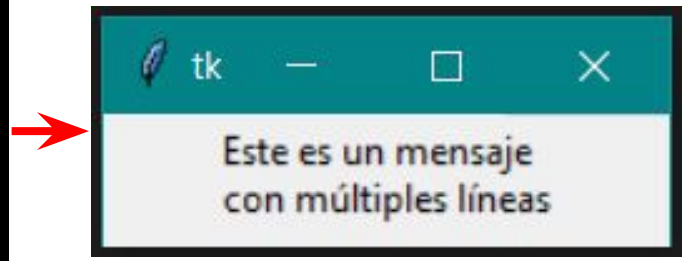
```
import tkinter as tk

ventana = tk.Tk()
# Crear una ventana dividida
panedwindow = tk.PanedWindow(ventana)
panedwindow.pack(fill=tk.BOTH, expand=1)

# Añadir panel izquierdo con un marco para poder visualizarlo
izquierda = tk.Label(panedwindow, text="Panel Izquierdo", bd=1,
relief="solid"))
panedwindow.add(izquierda)

# Añadir panel derecho con un marco para poder visualizarlo
derecha = tk.Label(panedwindow, text="Panel Derecho", bd=1,
relief="solid"))
panedwindow.add(derecha)

ventana.mainloop()
```



6.5 Massage

Muestra un mensaje de texto que puede contener múltiples líneas.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un mensaje con múltiples líneas
mensaje = tk.Message(ventana, text="Este es un mensaje\ncon múltiples
líneas", width=200)
mensaje.pack()  # Colocar el mensaje en la ventana
ventana.mainloop()
```



6.6 Entry

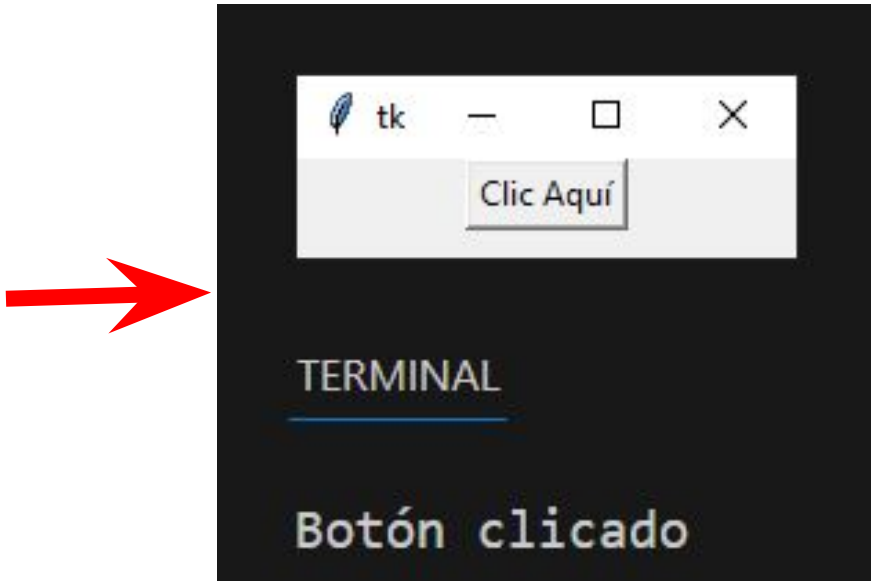
Permite al usuario ingresar y editar una línea de texto de una sola línea.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un campo de entrada de texto
entrada = tk.Entry(ventana)
entrada.pack()  # Colocar el campo de entrada en la ventana
ventana.mainloop()
```

6.7 Button

Es un widget que permite al usuario interactuar con la interfaz haciendo clic en él para activar alguna acción.

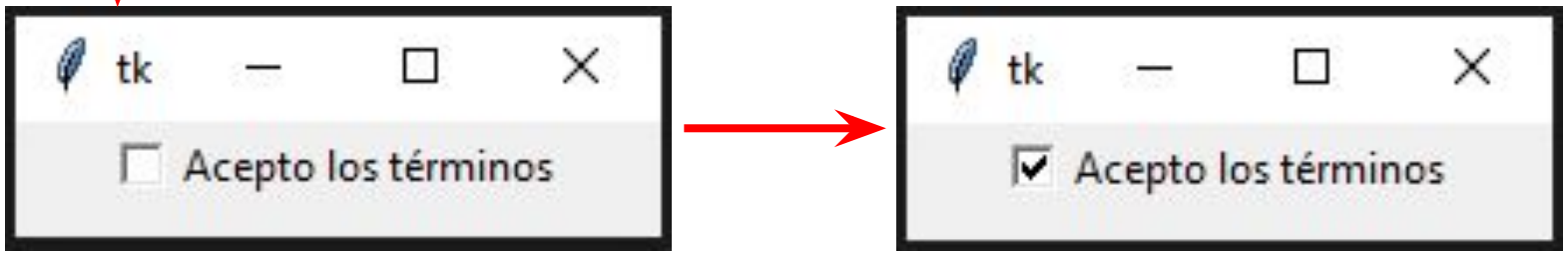
```
import tkinter as tk

def accion():
    print("Botón clicado")

ventana = tk.Tk()
# Crear un botón con el texto "Clic Aquí" que llama a la función 'accion' cuando se clickea
boton = tk.Button(ventana, text="Clic Aquí", command=accion)
boton.pack()  # Colocar el botón en la ventana
ventana.mainloop()
```



6.8 Checkbutton

Es una casilla que puede ser marcada o desmarcada por el usuario.

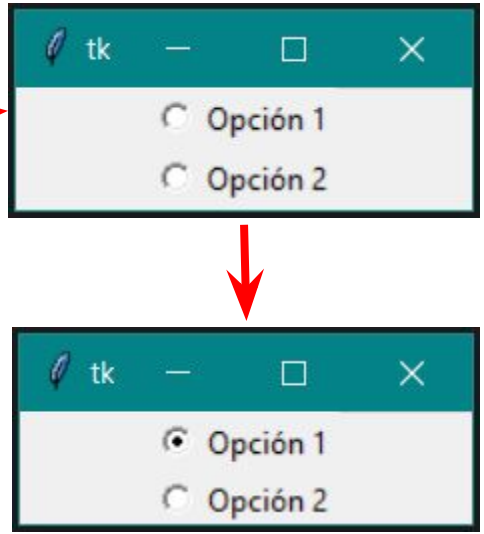
```
import tkinter as tk

ventana = tk.Tk()
var = tk.IntVar()  # Variable que mantiene el estado de la casilla
# Crear una casilla con el texto "Acepto los términos"
casilla = tk.Checkbutton(ventana, text="Acepto los términos", variable=var)
casilla.pack()  # Colocar la casilla en la ventana
ventana.mainloop()
```



6.9 Radiobutton

Permite al usuario seleccionar una opción entre varias opciones mutuamente excluyentes.

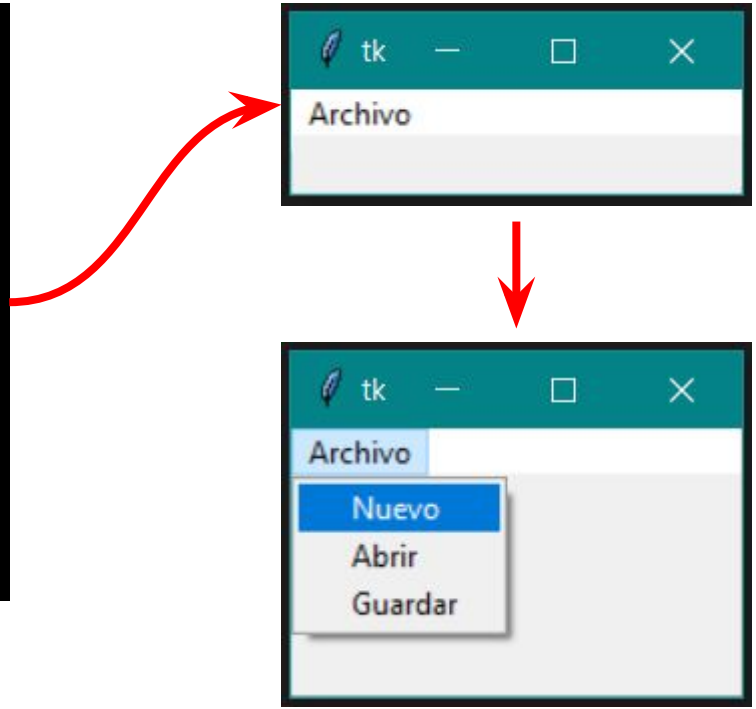
```
import tkinter as tk

ventana = tk.Tk()
var = tk.IntVar()  # Variable que mantiene la opción seleccionada
# Crear opciones de botones de radio
radio1 = tk.Radiobutton(ventana, text="Opción 1", variable=var, value=1)
radio2 = tk.Radiobutton(ventana, text="Opción 2", variable=var, value=2)
radio1.pack()  # Colocar los botones de radio en la ventana
radio2.pack()
ventana.mainloop()
```



6.10 Menu

Representa un menú desplegable que puede contener varios elementos de menú.

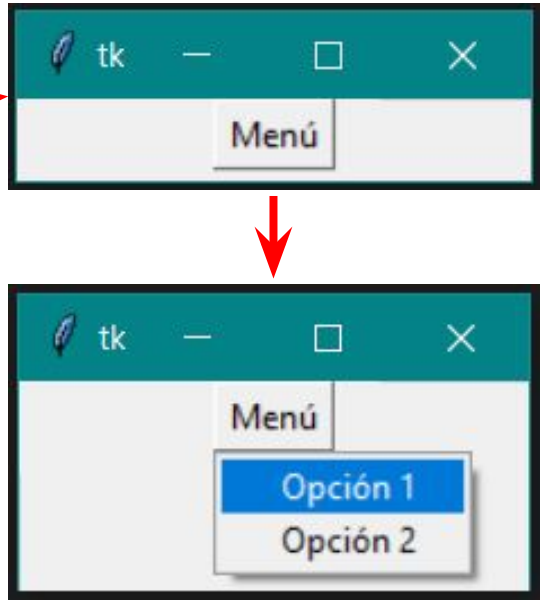
```
import tkinter as tk

ventana = tk.Tk()
menubar = tk.Menu(ventana)
archivo = tk.Menu(menubar, tearoff=0)
archivo.add_command(label="Nuevo")  # Añadir opciones al menú
archivo.add_command(label="Abrir")
archivo.add_command(label="Guardar")
menubar.add_cascade(label="Archivo", menu=archivo)  # Añadir el menú a la barra de menús
ventana.config(menu=menubar)  # Configurar la ventana para usar la barra de menús
ventana.mainloop()
```



6.11 Menubutton

Muestra un menú desplegable cuando es clicado.

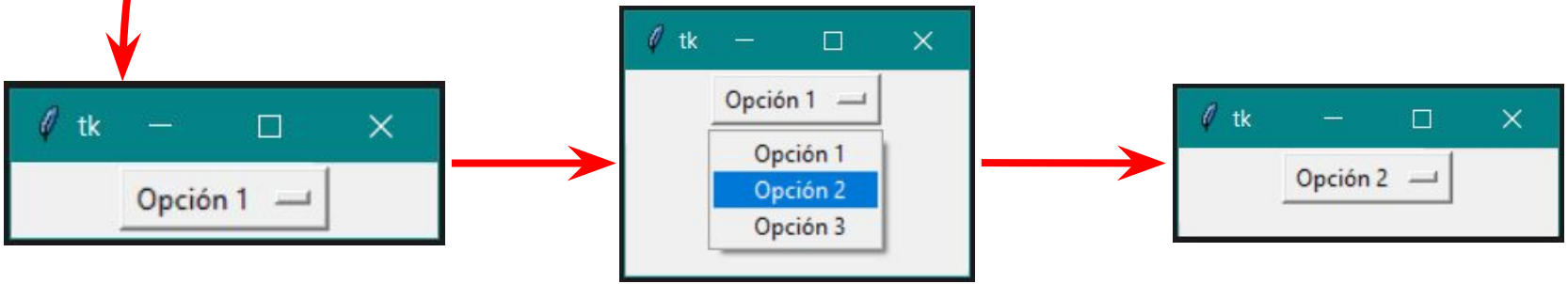
```
import tkinter as tk

ventana = tk.Tk()
# Crear un botón de menú
boton_menu = tk.Menubutton(ventana, text="Opciones", relief="raised")
menu = tk.Menu(boton_menu, tearoff=0)
boton_menu["menu"] = menu
menu.add_command(label="Opción 1")  # Añadir opciones al menú
menu.add_command(label="Opción 2")
boton_menu.pack()  # Colocar el botón de menú en la ventana
ventana.mainloop()
```



6.12 OptionMenu

Muestra un menú desplegable que permite al usuario seleccionar una opción de una lista predefinida.

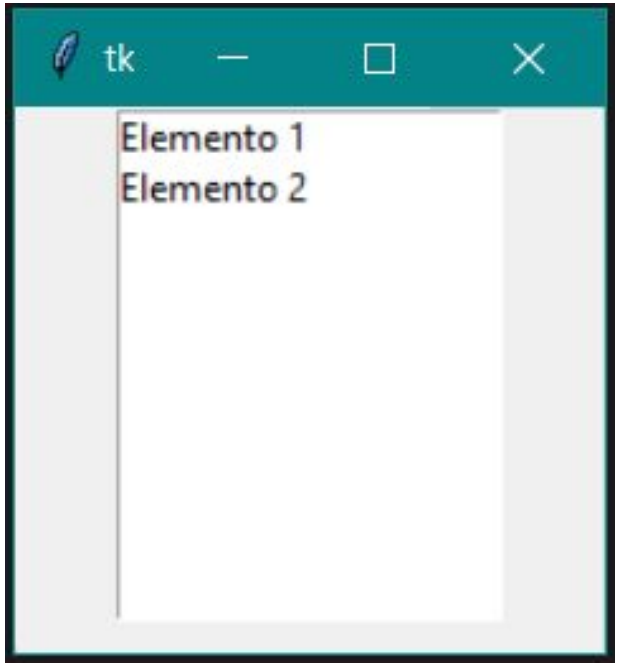
```
import tkinter as tk

ventana = tk.Tk()
opcion = tk.StringVar(ventana)  # Variable que mantiene la opción
seleccionada
opcion.set("Opción 1")  # Opción por defecto
# Crear un menú de opciones
menu_opciones = tk.OptionMenu(ventana, opcion, "Opción 1", "Opción 2",
"Opción 3")
menu_opciones.pack()  # Colocar el menú de opciones en la ventana
ventana.mainloop()
```



6.13 Listbox

Muestra una lista de elementos de texto, de los cuales el usuario puede seleccionar uno o más.

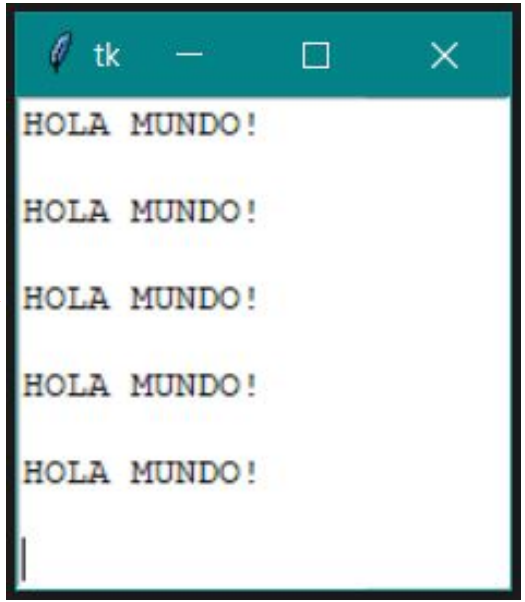
```
import tkinter as tk

ventana = tk.Tk()
# Crear una lista
lista = tk.Listbox(ventana)
lista.insert(1, "Elemento 1")  # Insertar elementos en la lista
lista.insert(2, "Elemento 2")
lista.pack()  # Colocar la lista en la ventana
ventana.mainloop()
```



6.14 Text

Permite al usuario ingresar y editar texto con múltiples líneas.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un campo de texto de múltiples líneas
texto = tk.Text(ventana)
texto.pack(expand=True, fill='both')  # Colocar el campo de texto en la ventana
ventana.mainloop()
```



6.15 Scrollbar

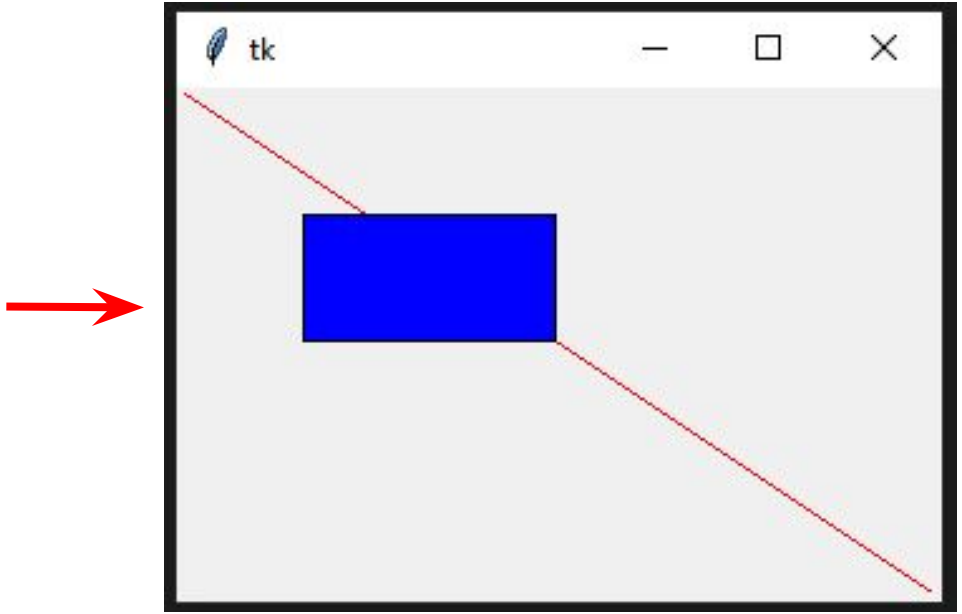
Permite añadir barras de desplazamiento a otros widgets, como Text, Canvas o Listbox.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un campo de texto
texto = tk.Text(ventana, wrap=tk.WORD)
texto.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

# Crear una barra de desplazamiento
scroll = tk.Scrollbar(ventana, command=texto.yview)
scroll.pack(side=tk.RIGHT, fill=tk.Y)
texto.config(yscrollcommand=scroll.set)  # Enlazar la barra de
desplazamiento con el campo de texto
ventana.mainloop()
```

6.16 Canvas

Actúa como un lienzo en el que se pueden dibujar formas, líneas, imágenes y texto.

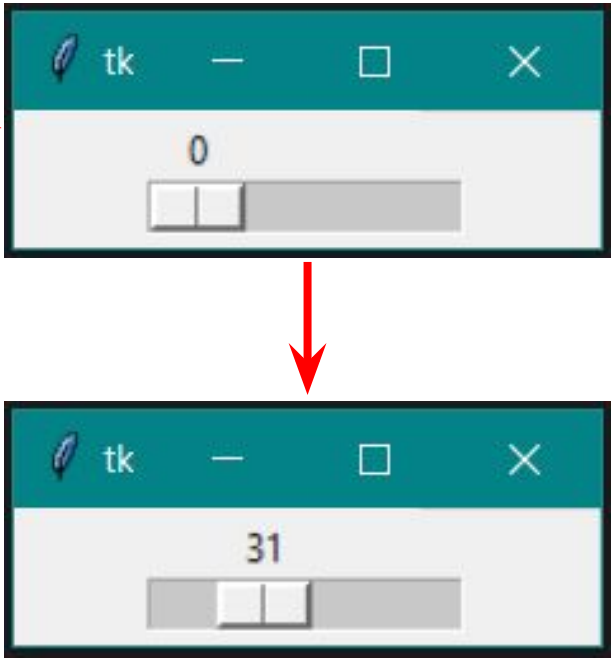
```
import tkinter as tk

ventana = tk.Tk()
# Crear un lienzo de 300x200 píxeles
lienzo = tk.Canvas(ventana, width=300, height=200)
lienzo.pack()  # Colocar el lienzo en la ventana
# Dibujar una línea roja desde (0,0) a (300,200)
lienzo.create_line(0, 0, 300, 200, fill="red")
# Dibujar un rectángulo azul
lienzo.create_rectangle(50, 50, 150, 100, fill="blue")
ventana.mainloop()
```



6.17 Scale

Permite al usuario seleccionar un valor de un rango deslizante.

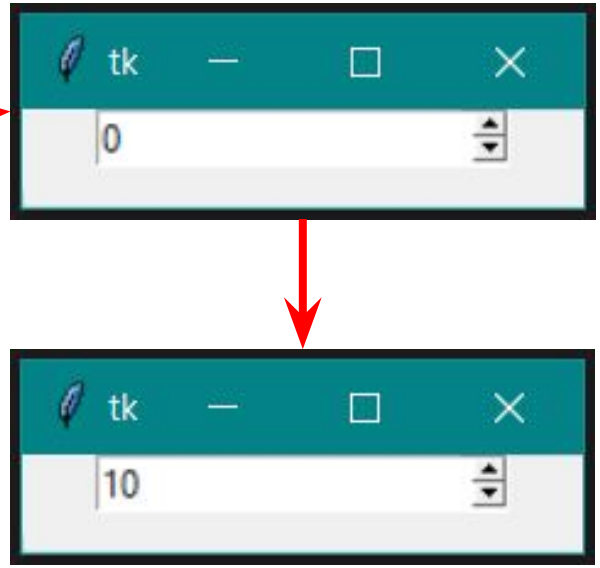
```
import tkinter as tk

ventana = tk.Tk()
# Crear un deslizador de valores de 0 a 100
escala = tk.Scale(ventana, from_=0, to=100, orient=tk.HORIZONTAL)
escala.pack()  # Colocar el deslizador en la ventana
ventana.mainloop()
```



6.18 Spinbox

Permite al usuario seleccionar un valor de una lista predefinida de valores.

```
import tkinter as tk

ventana = tk.Tk()
# Crear un cuadro de selección de valores de 0 a 10
spinbox = tk.Spinbox(ventana, from=0, to=10)
spinbox.pack()  # Colocar el cuadro de selección en la ventana
ventana.mainloop()
```



7. Posicionamiento de Widgets

Métodos de Posicionamiento:

1. **pack():** Posicionamiento relativo. Coloca los widgets uno después del otro.

2. **grid():** Sistema de grillas. Permite ubicar widgets en una estructura de filas y columnas.

3. **place():** Posicionamiento absoluto. Ubica los widgets en coordenadas específicas.

7.1 Ejemplo del Posicionamiento Relativo: pack()

```
import tkinter as tk

# Crear la ventana principal
root = tk.Tk()
root.title("Ejemplo de pack()")

# Crear una etiqueta y posicionarla usando pack()
label1 = tk.Label(root, text="Etiqueta 1", bg="red")
label1.pack(fill=tk.X, padx=10, pady=5)

# Crear un botón y posicionarlo usando pack()
button1 = tk.Button(root, text="Botón 1", bg="blue", fg="white")
button1.pack(fill=tk.X, padx=10, pady=5)

# Crear otra etiqueta y posicionarla usando pack()
label2 = tk.Label(root, text="Etiqueta 2", bg="green", fg="white")
label2.pack(side=tk.LEFT, fill=tk.Y, padx=5, pady=10)

# Crear un campo de entrada y posicionarlo usando pack()
entry1 = tk.Entry(root)
entry1.pack(side=tk.LEFT, padx=5, pady=10)

# Mostrar la ventana
root.mainloop()
```

7.2 Ejemplo del Posicionamiento en Forma de Grilla: grid()

```
import tkinter as tk

# Crear la ventana principal
root = tk.Tk()
root.title("Ejemplo de grid()")

# Crear una etiqueta y posicionarla usando grid()
label1 = tk.Label(root, text="Fila 0, Columna 0", bg="red", fg="white", padx=10, pady=10)
label1.grid(row=0, column=0, padx=5, pady=5)

# Crear una etiqueta y posicionarla en la misma fila pero en la siguiente columna
label2 = tk.Label(root, text="Fila 0, Columna 1", bg="blue", fg="white", padx=10, pady=10)
label2.grid(row=0, column=1, padx=5, pady=5)

# Crear un botón y posicionarlo en la siguiente fila
button1 = tk.Button(root, text="Fila 1, Columna 0", bg="green", fg="white", padx=10, pady=10)
button1.grid(row=1, column=0, padx=5, pady=5)

# Crear otra etiqueta y hacer que ocupe dos columnas
label3 = tk.Label(root, text="Fila 1, Columna 1 y 2", bg="yellow", fg="black", padx=10, pady=10)
label3.grid(row=1, column=1, columnspan=2, padx=5, pady=5)

# Crear un botón y hacerlo ocupar dos filas
button2 = tk.Button(root, text="Fila 2 y 3, Columna 0", bg="purple", fg="white", padx=10, pady=10)
button2.grid(row=2, column=0, rowspan=2, padx=5, pady=5)

# Crear una etiqueta y posicionarla en la tercera fila y segunda columna
label4 = tk.Label(root, text="Fila 2, Columna 1", bg="orange", fg="black", padx=10, pady=10)
label4.grid(row=2, column=1, padx=5, pady=5)

# Mostrar la ventana
root.mainloop()
```

7.3 Ejemplo de Posición Absoluta: place()

```
import tkinter as tk

# Crear la ventana principal
root = tk.Tk()
root.title("Ejemplo de place()")

# Crear una etiqueta y posicionarla usando place()
label1 = tk.Label(root, text="Etiqueta 1", bg="red")
label1.place(x=50, y=50)

# Crear un botón y posicionarlo usando place()
button1 = tk.Button(root, text="Botón 1", bg="blue", fg="white")
button1.place(x=150, y=100)

# Crear otra etiqueta con tamaño específico y posicionarla usando place()
label2 = tk.Label(root, text="Etiqueta 2", bg="green", fg="white")
label2.place(x=200, y=200, width=100, height=50)

# Crear un campo de entrada y posicionarlo usando place()
entry1 = tk.Entry(root)
entry1.place(x=100, y=150)

# Mostrar la ventana
root.mainloop()
```

8. Modificación de Widgets

Opciones Comunes:

- **bg:** Color de fondo
- **fg:** Color de la fuente
- **bd:** Ancho del borde en píxeles
- **font:** Tipo de fuente
- **relief:** Especifica el tipo de borde. Algunos de los valores son SUNKEN, RAISED, GROOVE y RIDGE
- **bitmap:** Reemplaza el texto por uno de los bitmaps disponibles (error, gray75, gray50, gray25, gray12, hourglass, info, questhead, question, warning)
- **width/height:** Ancho y alto en caracteres
- **image:** Imagen en lugar de texto
- **justify:** Justificación del texto
- **state:** Permite indicar si el widget está en modo NORMAL, DISABLED o ACTIVE

8.1 Ejemplos de Uso

```
from tkinter import Tk, Label, Button

# Crear una ventana principal
root = Tk()

# Crear una etiqueta con texto alineado a la izquierda, fondo azul y espaciado
label1 = Label(root, text="Fila 0, Columna 0", fg='white', bg='blue', padx=10, pady=10)
label1.grid(row=0, column=0, padx=10, pady=10)

# Crear una etiqueta con texto alineado a la izquierda, fondo amarillo y tamaño específico
label2 = Label(root, text="Fila 0, Columna 1", fg='black', bg='yellow', width=30, anchor='w')
label2.grid(row=0, column=1)

# Crear un botón con fondo verde oscuro y texto claro
boton1 = Button(root, text='¡Haz clic!', padx=15, pady=15, bg='dark green', fg='light green')
boton1.grid(row=0, column=2, padx=10, pady=10)

# Crear una etiqueta con borde de 15 píxeles y relieve en forma de 'groove'
label3 = Label(root, text="Fila 1, Columna 0", bg='red', padx=5, pady=5, bd=15, relief='groove')
label3.grid(row=1, column=0, padx=10, pady=10)

# Crear un botón con fuente 'Courier New' tamaño 15 y fondo activo magenta
boton2 = Button(root, text="¡Haz clic!", font=("Courier New", 15), activebackground='magenta')
boton2.grid(row=1, column=1, padx=18, pady=10)

# Cambiar el cursor a forma de 'spider' cuando esté sobre el botón
boton2.config(cursor="spider")

# Mostrar la ventana
root.mainloop()
```

Aquí cambiamos el color de fondo (`bg`), el color del texto (`fg`) y la fuente (`font`) del `Label` para personalizar su apariencia.

9. Uso de MessageBox

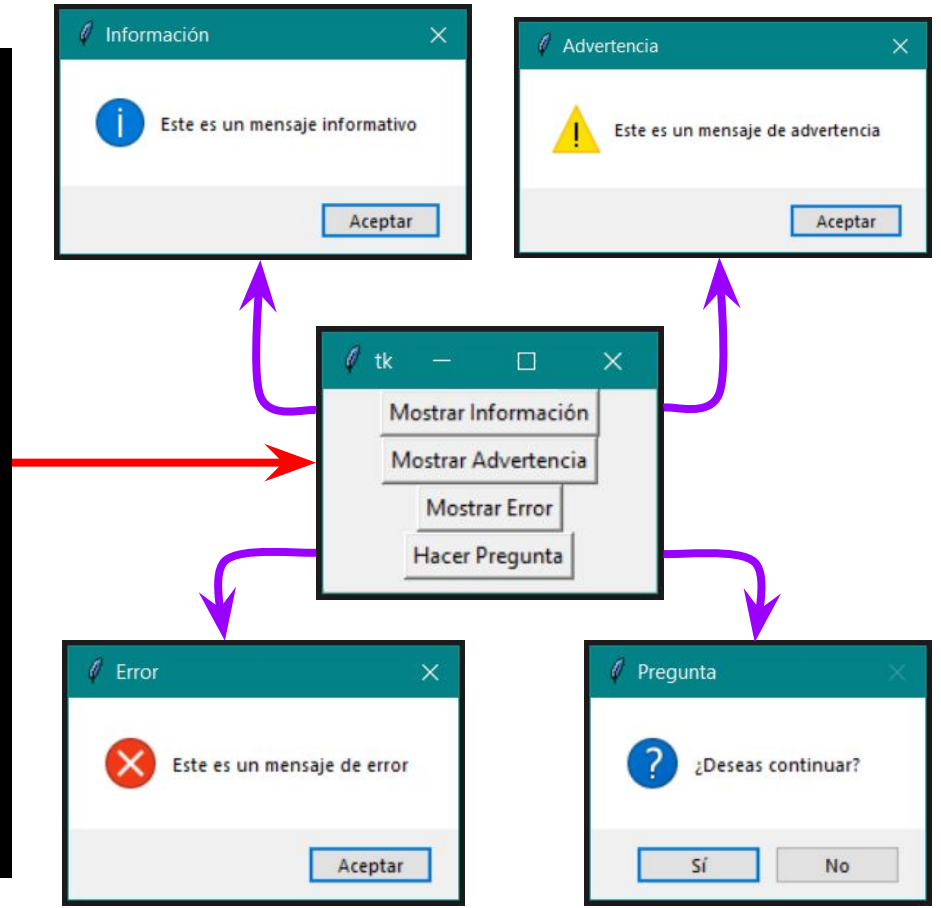
MessageBox: es una ventana emergente utilizada para mostrar mensajes al usuario. Es una herramienta esencial para informar, advertir, mostrar errores o solicitar confirmaciones.

Características Clave:

- **Tipos de MessageBox:**
  - **showinfo:** Muestra un mensaje informativo.
  - **showwarning:** Muestra una advertencia.
  - **showerror:** Muestra un mensaje de error.
  - **askquestion:** Pregunta con opciones "Yes" y "No".
  - **askokcancel:** Pregunta con opciones "OK" y "Cancel".
  - **askyesno:** Pregunta con opciones "Yes" y "No".
  - **askretrycancel:** Pregunta con opciones "Retry" y "Cancel"

9.1 Ejemplos de Uso

```
import tkinter as tk
from tkinter import messagebox

# Crear la ventana principal
root = tk.Tk()

# Funciones para mostrar MessageBox
def mostrar_info():
    messagebox.showinfo("Información", "Este es un mensaje informativo")

def mostrar_advertencia():
    messagebox.showwarning("Advertencia", "Este es un mensaje de advertencia")

def mostrar_error():
    messagebox.showerror("Error", "Este es un mensaje de error")

def preguntar():
    respuesta = messagebox.askquestion("Pregunta", "¿Deseas continuar?")
    print("Respuesta:", respuesta)

# Crear botones para mostrar MessageBox
tk.Button(root, text="Mostrar Información", command=mostrar_info).pack()
tk.Button(root, text="Mostrar Advertencia", command=mostrar_advertencia).pack()
tk.Button(root, text="Mostrar Error", command=mostrar_error).pack()
tk.Button(root, text="Hacer Pregunta", command=preguntar).pack()

# Mostrar la ventana
root.mainloop()
```



10. Trabajando con Imágenes

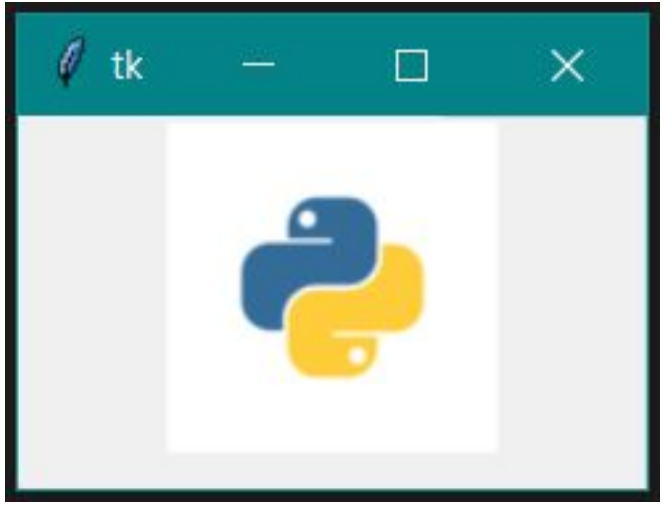
PhotoImage : Se utiliza para trabajar con imágenes. Permite cargar, manipular y mostrar imágenes en aplicaciones GUI construidas con Tkinter. Es compatible con formatos de imagen como GIF, PGM y PPM.

Características Clave:

- **Carga de Imágenes:**
  - PhotoImage permite cargar imágenes desde archivos y datos en formato de cadena.
- **Compatibilidad:**
  - Compatible con formatos GIF, PGM y PPM.
- **Uso Común:**
  - Utilizado para mostrar imágenes en widgets como Label, Button, Canvas, etc.

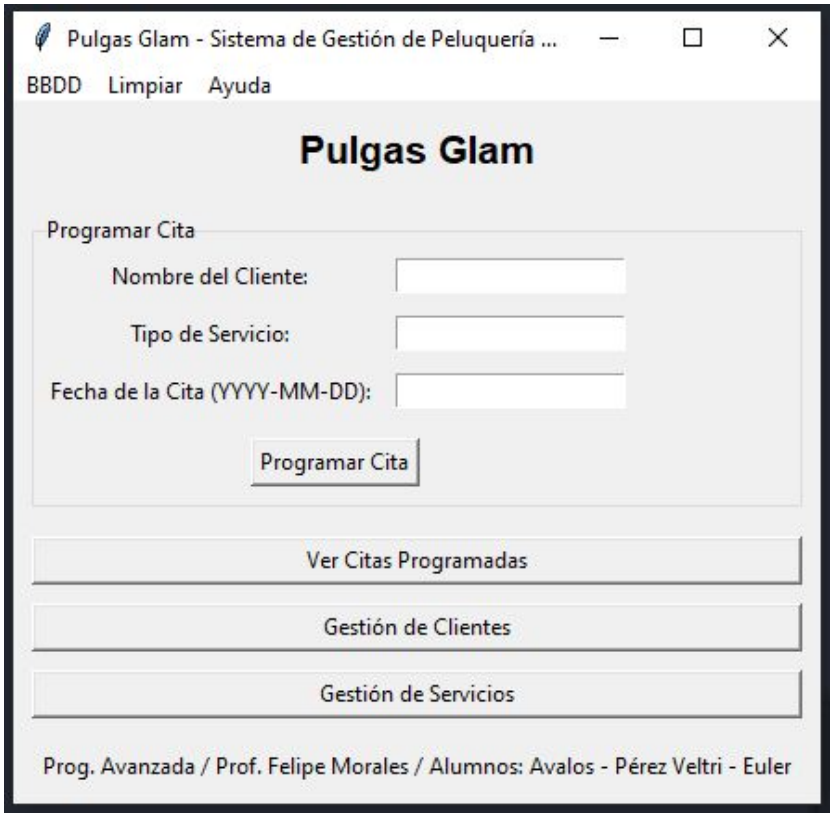
10.1 Ejemplos de Uso

```
import tkinter as tk
from tkinter import PhotoImage

# Crear una ventana principal
root = tk.Tk()

# Cargar una imagen desde un archivo (asegúrate de tener "imagen.png" en la misma carpeta)
img = PhotoImage(file="imagen.png")

# Crear una etiqueta y asociar la imagen
label = tk.Label(root, image=img)
label.pack()

# Mostrar la ventana
root.mainloop()
```



11. Desarrollo de aplicaciones robustas y atractivas

Pulgas Glam - Sistema de Gestión de Peluquería Canina



Documentación y Recursos Adicionales

Recursos para Aprender Más

- Documentación oficial: docs.python.org
- Tutoriales: tkdocs.com
- Ejemplos y Guías: realpython.com

Conclusión

- Tkinter es una herramienta poderosa y fácil de usar para crear GUIs en Python.
- Permite la creación de interfaces interactivas y personalizables.
- Con las técnicas y ejemplos mostrados, se pueden desarrollar aplicaciones robustas y atractivas.

¡Gracias por su atención!

unab