

Apunte Tkinter

27/10/2025

1.- Pesar Argumentos al comando del botón Tkinter

Para que el botón ejecute una función que requiere argumentos, no se puede llamar directamente en la opción `command`, ya que esto ejecutará la función inmediatamente al crear el botón. La solución es extender la función `lambda`, la cual crea una pequeña función anónima que encapsula la llamada a nuestra función principal, siendo de manera usual:

```
import tkinter as tk
def saludar(nombre):
    print(f"Hello, {Nombre}")

# Crear la Ventana (usando)
# Ventana = tk.Tk()

# Al crear el botón, usamos lambda para retrasar la acción
# y pasar el argumento Mundo a la función saludar
boton = tk.Button(
    text="Saludar",
    command=lambda: saludar("Mundo")
)
boton.pack(pady=5)
```

Caso 1

17/10/2025

2.- Cerrar una ventana de Tkinter con un botón

Cada ventana de Tkinter (tanto la raíz `tk.Tk()` como las ventanas secundarias `tk.Toplevel()`) posee un método llamado `.destroy()`. Siendo este de cerrar y limpiar los recursos asociados a ella. Para realizar que un botón cierre la ventana que lo contiene se le asigna el método `.destroy()`:

cerrar = `tk.Button(ventana,`

`text = "Cerrar ventana"`

`command = ventana.destroy`

)

botón cerrar

`cerrar.pack(pady=5)`

Caso 2

3.- Cambiar el estado del Botón

Los botones tienen la propiedad `state`, los estados más comunes son `normal` y `disabled`. Podemos cambiar este estado directamente después de crear el botón, se utiliza el método `.config(state=...)`:

def desabilitar_boton():

`boton_estado.config(state="disabled")`

def habilitar_boton():

`boton_estado.config(state="normal")`

4.- Crear una nueva ventana

haciendo click en un Botón

Tkinter proporciona el widget `Toplevel`, `boton_estado = tk.Button(ventana, text="click")` Este crea una ventana que depende de la para deshabilitar!, `command = desabilitar_boton` Ventana principal (Si se cierra la principal, se cierra la secundaria):

`boton_estado.pack(pady=5)`

Caso 3

def ventana_nueva():

`secundaria = tk.Toplevel(ventana)`

`secundaria.title("Ventana Secundaria")`

`secundaria.geometry("200x100")`

`tk.Label(secundaria, text = "Otra ventana").pack(pady=20)`

`abrir = tk.Button(`

`ventana,`

`text = "Nueva Ventana"`

`command = abrir_ventana_nueva`

`)`

`boton_abrir.pack(pady=5)`

Caso 4

5.- Vincular varios comandos a un botón

La opción `command` por si sola hace que solo pueda aceptar una única referencia a una función. Si de un solo click realizase varias funciones distintas, se debe crear una función contenedora en `wrapper`, que llamará una sucesión de todas las funciones que se deseen ejecutar:

```
def acción_1():
    print("ejecuto acción 1")
```

```
def acción_2():
    print("ejecuto acción 2")
```

```
def ejecutar_múltiples_comandos():
    acción_1()
    acción_2()
```

```
print("Ambas acciones completadas")
```

```
boton_multiple = tk.Button(
```

```
ventana,
```

```
text="Ejecutar Múltiples"
```

```
command=ejecutar_múltiples_comandos
```

```
)
```

```
boton_multiple.pack(pady=5)
```

Caso 5

6.- Cambiar el tamaño del botón

Este se puede controlar con las opciones `width` (ancho) y `height` (alto). Estas ~~no~~ son pixels; si no unidades de texto. Y si se desea un control de pixeles, a medida se debe usar `france` ya sea un tamaño fijo.

```
Tamaño = tk.Button(
```

```
ventana,
```

```
text="Botón Grande",
```

```
width=25,
```

```
height=3
```

```
).Tamaño.pack(pady=5)
```

Caso 6

7.- Actualizar el texto del botón

Existen dos formas principales de hacerlo después de crearlos. La más directa es usando .config() y reasignar la opción text. La segunda forma consiste en usar una variable de control StringVar. Se asigna esta variable a la opción "textvariable" del botón, y cada vez que se cambie el valor, el texto del botón se actualiza automáticamente.

Método 1 (Usando .config())

```
def cambiar_texto():
    boton.config(text="Cambio")
boton.config = tk.Button(ventana, text="Original text", command=cambiar_texto)
boton.config.pack(pady=5)
```

Método 2 (Usando StringVar)

```
text_button = tk.StringVar()
text_button.set("Otro texto")
def cambiar_texto_var():
    texto_dcl_button.set("Actualizado")
```

```
boton_stringvar = tk.Button(ventana, textvariable=text_button, command=cambiar_texto_var)
boton_stringvar.pack(pady=5)
```

Caso 7

27/10/2025

8.- Cambiar el color del botón

Esta se puede modificar con las opciones de configuración. Los más importantes son **bg** (background) para el color de fondo del botón, y **fg** (foreground) para el color del texto. Estos colores se pueden especificar al crear el botón o modificarse después usando `.config()`. Se pueden usar nombres de colores en inglés o en código hexadecimal:

```
def cambiar_color():
    boton_color.config(bg="blue", fg="white")
```

```
boton_color = tk.Button(
```

```
ventana,
```

```
text="Click para cambiar de color"
```

```
bg="#4CAF50",
```

```
fg="white"
```

```
command=cambiar_color
```

```
)
```

```
boton_color.pack(pady=5)
```

Caso 8