Interfaces Gráficas (GUI) con Tkinter

Clase 2: Interactividad y Controles

Prof. D.Sc. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

6 de noviembre de 2025

Agenda

Dando Vida a los Widgets

Repaso de la Clase 1

En la clase anterior, creamos una ventana con "widgets" (controles), pero eran **estáticos**. El botón estaba ahí, pero hacer clic en él no provocaba ninguna acción.

El Evento `command`

La forma más simple de hacer que un widget sea interactivo es a través de su propiedad `command`.

- La propiedad `command` se usa principalmente en los botones.
- Le decimos al botón: "Cuando alguien te haga clic, ejecuta esta función de Python".
- A esto se le llama **"callback"** (llamada de vuelta).

Conectando un Botón a una Función

Paso 1: Definir la Función

Primero, creamos una función de Python (con `def`) que contenga el código que queremos ejecutar.

```
1 def decir_hola():
2     print("¡Hola! Has hecho clic en el boton.")
3
```

Conectando un Botón a una Función

Paso 2: Conectar el Botón con `command`

Al crear el botón, le asignamos el **nombre de la función** a su propiedad `command`.

```
# ¡MUY IMPORTANTE!
# Se pasa el nombre de la funcion SIN parentesis:
#
# Correcto:   command=decir_hola
# Incorrecto: command=decir_hola()

boton = tk.Button(ventana,
                  text="Haz Clic",
                  command=decir_hola)

```

Agenda

Control de Entrada: `tk.Entry`

¿Qué es un Widget `Entry`?

Es el widget estándar para que el usuario pueda **ingresar una sola línea de texto** (como un nombre, una contraseña o un número). Es el equivalente gráfico de la función `input()` de la consola.

Control de Entrada: `tk.Entry`

Creación del Widget

Se crea de la misma forma que un `Label` o `Button`.

```
1  import tkinter as tk
2  ventana = tk.Tk()
3
4  # Crear un campo de entrada
5  etiqueta_nombre = tk.Label(ventana, text=''Nombre:'')
6  entrada_nombre = tk.Entry(ventana)
7
8  etiqueta_nombre.pack()
9  entrada_nombre.pack()
10
11 ventana.mainloop()
12
```

Obteniendo el Texto: El Método `.get()`

¿Cómo leemos lo que el usuario escribió?

El widget `Entry` tiene un método especial llamado `.get()` que nos permite **extraer** el texto que contiene en ese momento.

Obteniendo el Texto: El Método .get()

Uso de .get()

La llamada a .get() se hace típicamente **dentro de la función** que es llamada por el botón.

```
1  def on_button_click():
2      nombre_usuario = entrada_nombre.get()
3
4      print(f''El usuario escribio: {nombre_usuario}'')
5
6
7  boton = tk.Button(ventana,
8                    text=''Enviar'',
9                    command=on_button_click)
10
```

Agenda

Modificando Widgets: El Método `.config()`

El Problema

Hacer `print()` en la consola no es una verdadera aplicación gráfica. Queremos que la propia ventana **reaccione y cambie**.

Modificando Widgets: El Método `.config()`

La Solución: `.config()`

Todos los widgets tienen un método `.config()` que nos permite **cambiar sus propiedades** (como el texto) **después** de que han sido creados.

```
1
2  def on_button_click():
3      # Cambiamos el texto de 'etiqueta_saludo'
4      etiqueta_saludo.config(text="¡Has hecho clic!")
5
6  # Creamos una etiqueta...
7  etiqueta_saludo = tk.Label(ventana, text="Esperando clic...")
8  etiqueta_saludo.pack()
9
10 # ...y un boton que la controla
11 boton = tk.Button(ventana, command=on_button_click)
12 boton.pack()
13
```

Agenda

Proyecto: Aplicación "Saludador"

Objetivo

Combinar todo lo aprendido:

1. Un `Label` y un `Entry` para pedir un nombre.
2. Un `Button` para ejecutar la acción.
3. Un `Label` vacío que se actualizará con el saludo.

Proyecto: Aplicación "Saludador"

Código Completo

```
import tkinter as tk
def saludar():
    nombre = entrada_nombre.get()
    saludo_final = f"¡Hola, {nombre}!"
    etiqueta_resultado.config(text=saludo_final)
ventana = tk.Tk()
ventana.title("Saludador v1.0")
# --- Widgets ---
etiqueta_instruccion = tk.Label(ventana, text="Escribe tu nombre:")
entrada_nombre = tk.Entry(ventana)
boton_saludar = tk.Button(ventana, text="Saludar", command=saludar)
etiqueta_resultado = tk.Label(ventana, text="")
# --- Posicionamiento ---
etiqueta_instruccion.pack()
entrada_nombre.pack()
boton_saludar.pack()
```

Agenda

Tarea 2: Mini-Calculadora (Sumadora)

Objetivo

Crear una aplicación gráfica que funcione como una sumadora simple, aplicando todo lo visto en clase.

Tarea 2: Mini-Calculadora (Sumadora)

Instrucciones

Tu programa debe tener la siguiente interfaz:

- Una etiqueta `tk.Label` con el texto "Número 1:".
- Un campo de entrada `tk.Entry` para el primer número.
- Una etiqueta `tk.Label` con el texto "Número 2:".
- Un segundo campo de entrada `tk.Entry` para el segundo número.
- Un botón `tk.Button` con el texto "Sumar".
- Una etiqueta `tk.Label` (inicialmente vacía) donde se mostrará el resultado (ej: "Resultado: 15").

Tarea 2: Mini-Calculadora (Sumadora)

Pistas Clave

- El método `.get()` **siempre devuelve un string**.
- Deberás **convertir** esos strings a números (con `int()` o `float()`) antes de poder sumarlos.
- Deberás crear una función (ej: `def sumar():`) y conectarla al `command` del botón.