python

Aplicaciones de Escritorio con Tkinter

Turbo Código

02

Posicionando controles

Posicionar controles dentro de una ventana

Gestores de geometría

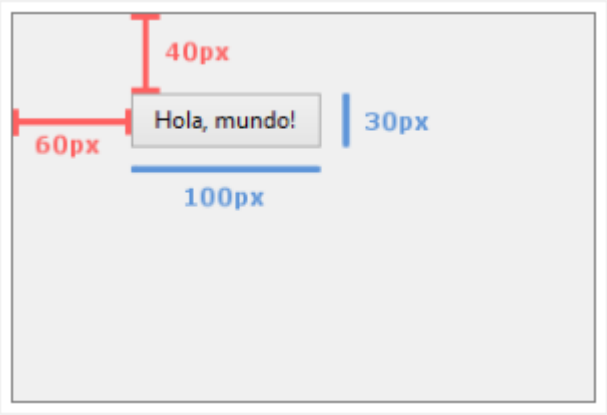
- Definir el **modo en que deben colocarse los** widgets (**controles**) dentro de una ventana.
  - place
  - pack
  - grid

- Para construir las ventanas se pueden utilizar unos widgets especiales (marcos, paneles, etc.) que **actúan como contenedores** de otros widgets. Estos widgets se utilizan para agrupar varios controles.
- No deben mezclarse distintos métodos dentro de un mismo contenedor.

Posición absoluta
Place

- Permite ubicar elementos indicando su posición (X e Y) respecto de un elemento padre.

```
self.button.place(x=60, y=40, width=100, height=30)
```



- Valores ABSOLUTOS

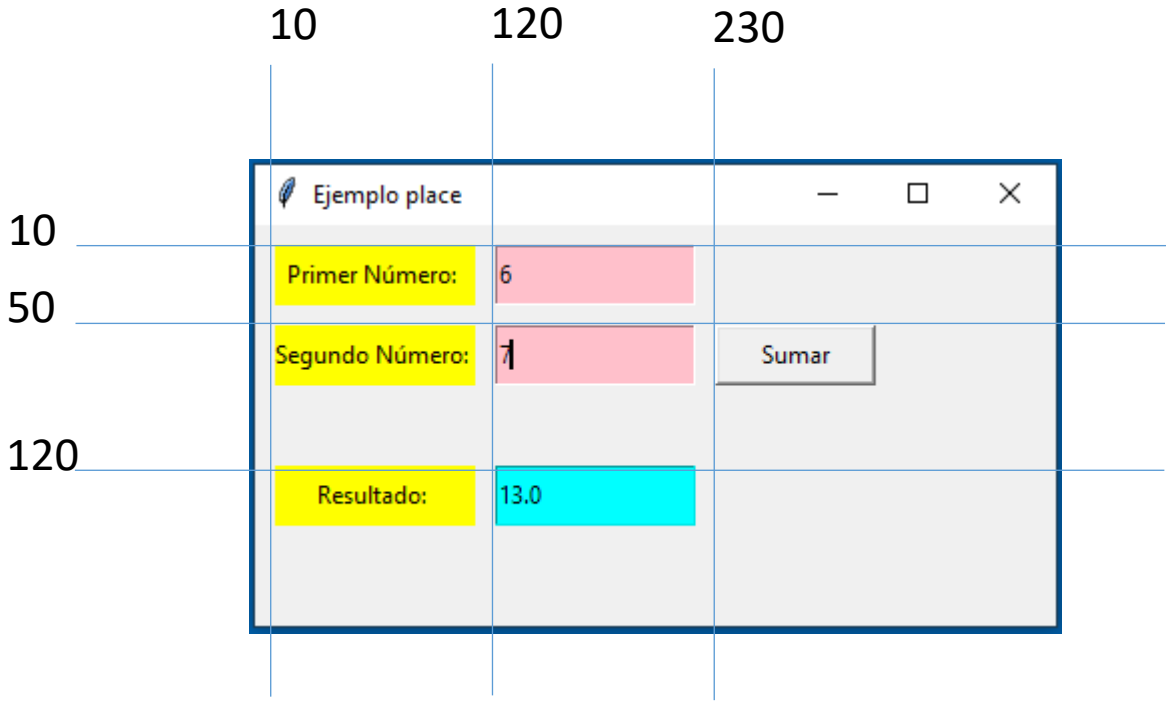
Posición absoluta
Place

- Permite ubicar elementos indicando su posición (X e Y) respecto de un elemento padre.

```
button.place(relx=0.1, rely=0.1, relwidth=0.5, relheight=0.5)
```

- Valores relativos al padre (contenedor)
- relwidth, relheight, relx y rely aceptan valores entre 0 y 1.

Ejemplo

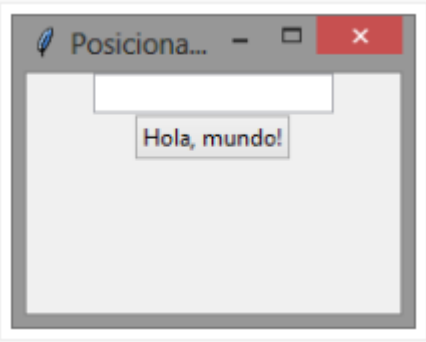


Turbo Código
https://www.youtube.com/user/juarezefren

Posicionamiento relativo (pack)

- En lugar de especificar las coordenadas de un elemento, simplemente le decimos que debe ir arriba, abajo, a la izquierda o a la derecha respecto de algún contenedor.
- Si no indicamos ningún argumento, por defecto Tk posicionará los elementos uno arriba del otro.

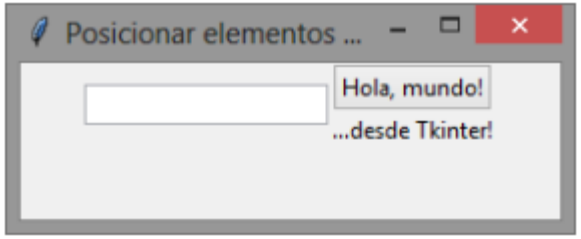
```
entry = ttk.Entry(self)
entry.pack()

button = ttk.Button(self, text="Hola, mundo!")
button.pack()
```



Posicionamiento relativo (pack)

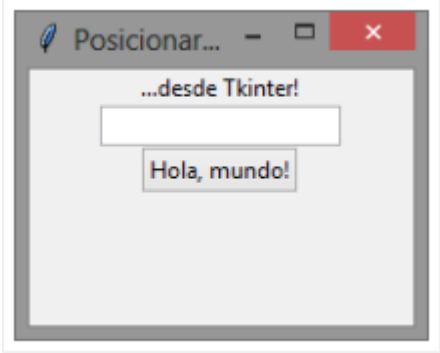
- La propiedad que controla la posición relativa de los elementos es side, que puede equivaler a tk.TOP (por defecto), tk.BOTTOM, tk.LEFT o tk.RIGHT. De este modo, si indicamos que la caja de texto debe ir ubicada a la izquierda, los otros dos controles se seguirán manteniendo uno arriba del otro.

```
self.entry = ttk.Entry(self)
self.entry.pack(side=tk.LEFT)
```



- También admite los parámetros after y before, que nos permiten controlar el orden en el que se ubican los elementos en la ventana. El siguiente ejemplo obliga a Tk a colocar la etiqueta self.label antes (before) que la caja de texto.

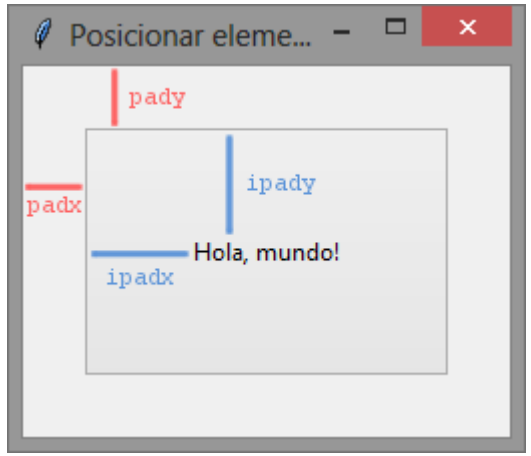
```
self.entry = ttk.Entry(self)
self.entry.pack()

self.button = ttk.Button(self, text="Hola, mundo!")
self.button.pack()

self.label = ttk.Label(self, text="...desde Tkinter!")
self.label.pack(before=self.entry)
```



Posicionamiento relativo (pack)

- **padx, ipadx, pady y ipady**
- Especifican (en píxeles) los margenes externos e internos de un elemento.

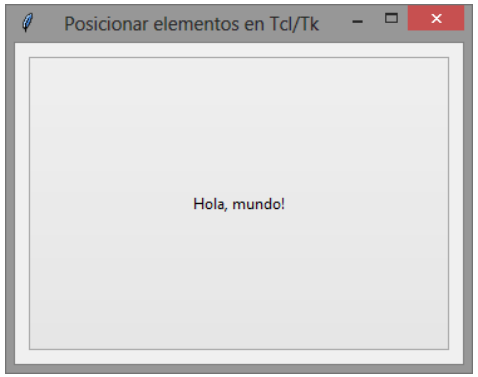
```
def __init__(self, main_window):
    super().__init__(main_window)
    main_window.title("Posicionar elementos en Tcl/Tk")

    self.button = ttk.Button(self, text="Hola, mundo!")
    self.button.pack(padx=30, pady=30, ipadx=50, ipady=50)

    self.pack()
```



Posicionamiento relativo (pack)

- **expand** True/False
- **fill** tk.BOTH, tk.X (horizontal), tk.Y (vetical)
- Especifican qué elementos deben expandirse o contraerse a medida que el tamaño de la ventana cambia, y en qué sentido deben hacerlo (vertical u horizontal),

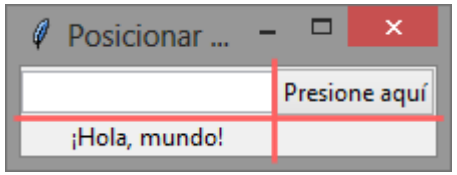
```
self.button = ttk.Button(self, text="Hola, mundo!")
self.button.pack(expand=True, fill=tk.X)

self.pack(expand=True, fill=tk.BOTH)
```



Manejo en forma de grilla (grid)

- Consiste en dividir conceptualmente la ventana principal en filas (rows) y columnas (columns), formando celdas en donde se ubican los elementos.

```
self.entry = ttk.Entry(self)
self.entry.grid(row=0, column=0)

self.button = ttk.Button(self, text="Presione aquí")
self.button.grid(row=0, column=1)

self.label = ttk.Label(self, text="¡Hola, mundo!")
self.label.grid(row=1, column=0)

self.grid(sticky="nsew")
```

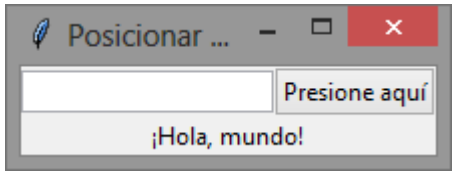


GRID	Column 0	Column 1	Colum 2
Row 0	Column 0 Row 0	Column 1 Row 0	Column 2 Row 0
Row 1	Column 0 Row 1	Column 1 Row 1	Column 2 Row 1
Row 2	Column 0 Row 2	Column 1 Row 2	Column 2 Row 2
Row 3	Column 0 Row 3	Column 1 Row 3	Column 2 Row 3

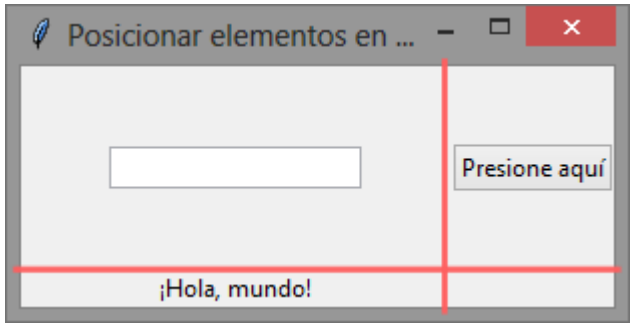
Manejo en forma de grilla (grid)

- **columnspan, rowspan:** Indica cuántas columnas o filas debe ocupar el control (por defecto 1)
- **rowconfigure**, **columnconfigure**: con el parámetro weight=1 indicamos que la fila o columna se expanden o contraen si la ventana cambia su tamaño.

```
self.label.grid(row=1, column=0, columnspan=2)
```



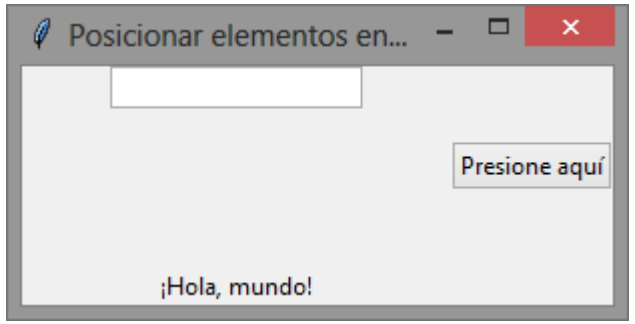

```
# Expandir horizontalmente a columna 0.
self.columnconfigure(0, weight=1)
# Expandir verticalmente la fila 0.
self.rowconfigure(0, weight=1)
```



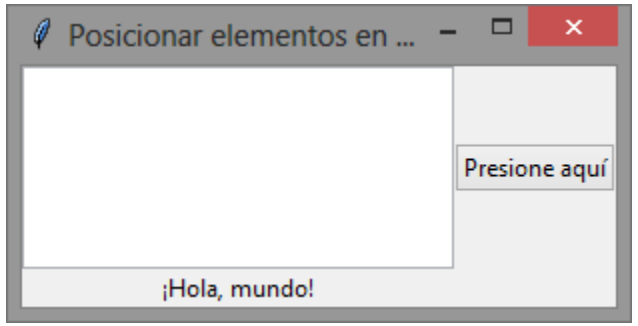
Manejo en forma de grilla (grid)

- **sticky**
- Para que un control se posicione arriba, abajo, a la derecha o izquierda de la celda que lo contiene, se usa sticky con las opciones: "n" (norte), "s" (sur), "e" (este) o "w" (oeste).
- También permite que el widget se expanda de forma horizontal ("ew"), vertical ("ns") o en ambas direcciones ("nsew").

```
# Anclar en la parte superior (n) de la celda.
self.entry.grid(row=0, column=0, sticky="n")
```



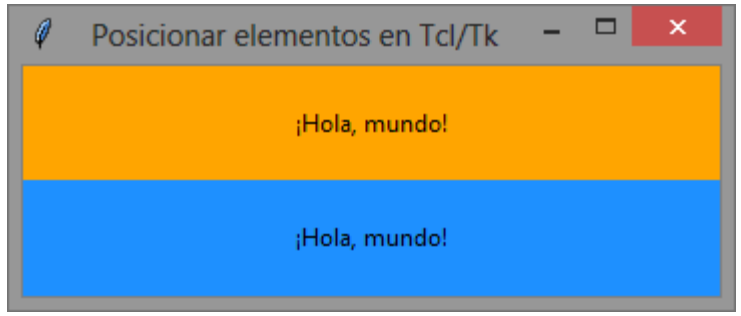

```
# Expandir en todas las direcciones.
self.entry.grid(row=0, column=0, sticky="nsew")
```



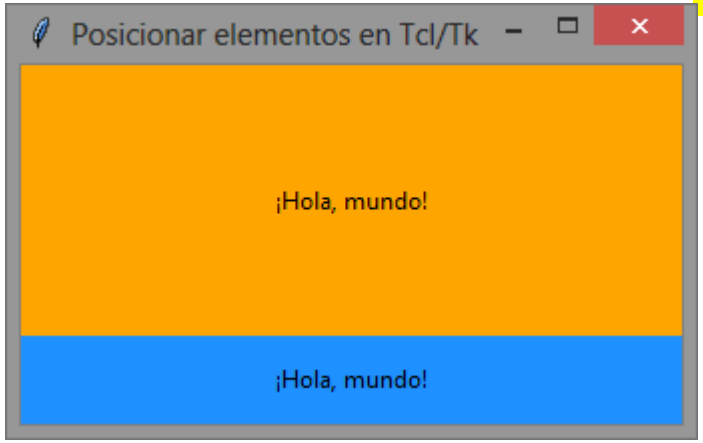
Manejo en forma de grilla (grid)

- **rowconfigure**, **columnconfigure**:

```
self.label1 = tk.Label(
    self, text="¡Hola, mundo!", bg="#FFA500")
self.label1.grid(row=0, column=0, sticky="nsew")

self.label2 = tk.Label(
    self, text="¡Hola, mundo!", bg="#1E90FF")
self.label2.grid(row=1, column=0, sticky="nsew")

self.grid(sticky="nsew")

self.columnconfigure(0, weight=1)
self.rowconfigure(0, weight=1)
self.rowconfigure(1, weight=1)
```



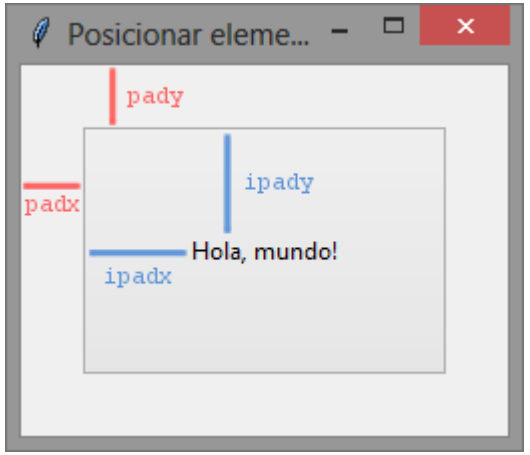

```
self.rowconfigure(0, weight=5)
self.rowconfigure(1, weight=1)
```



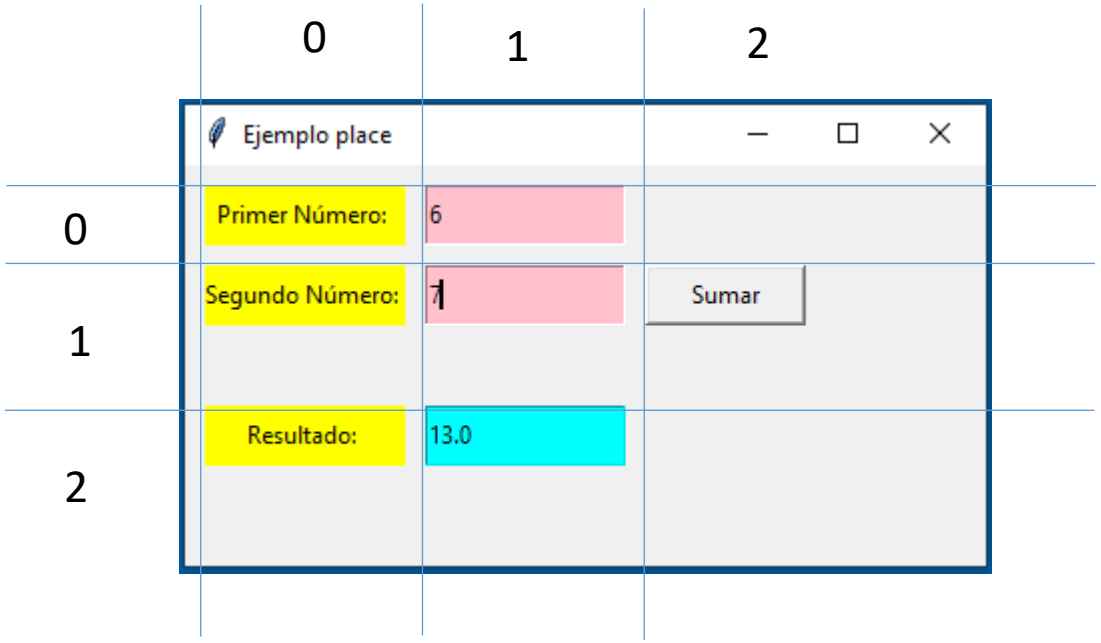
Manejo en forma de grilla (grid)

- **padx, ipadx, pady y ipady**
- Especifican (en píxeles) los margenes externos e internos de un elemento.

```
self.entry.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)
```



Ejemplo



Turbo Código
https://www.youtube.com/user/juarezefren

python

Aplicaciones de Escritorio con Tkinter

Turbo Código

02

Posicionando controles