

# 1. Bases de datos en Python: la librería sqlite3

## SQLite

Es una biblioteca que provee una base de datos liviana, basada en disco local, que no requiere procesos de servidor y permite acceder a los datos mediante SQL.

Se utiliza para almacenamiento interno y para prototipar aplicaciones, para luego transferir el código a una base de datos mayor.

## sqlite3

El módulo de Python **sqlite3** permite manejar bases de datos **SQLite 3.7.15** y superiores.

### Primeros pasos

En primer lugar, debemos importar el módulo y crear una conexión, que va a representar la base de datos. En el siguiente ejemplo, asignamos el alias **sq3** al módulo **sqlite3** y almacenamos los datos de nuestra base de datos en el archivo **mi\_db.db**.

Creamos una variable **con**, que representará nuestra base de datos.

```
1 import sqlite3 as sq3
2 con = sq3.connect('mi_db.db')
3 cur = con.cursor()
```

Una vez establecida la conexión, creamos un objeto de tipo **Cursor**, al que llamaremos **cur**, que invocará al método **execute()** para ejecutar los comandos SQL.

```
3 cur = con.cursor()
```

Ya estamos listos para preparar las instrucciones que van a crear una tabla, cargar y permitirnos visualizar el primer registro:

```
5 cur.execute('''CREATE TABLE IF NOT EXISTS
6 |         alumnos (legajo VARCHAR(7) PRIMARY KEY,
7 |         nombre VARCHAR(30),
8 |         apellido VARCHAR(30),
9 |         nota DECIMAL(10,0))''')
10
11 cur.execute('INSERT INTO alumnos VALUES ("2154448", "Ana", "Pérez", 8.5)')
12
13 for registro in cur.execute('SELECT * FROM alumnos'):
14     print(registro)
15
```

Ejecutamos estas instrucciones por medio del método **commit()** y cerramos la conexión con la base de datos mediante el método **close()**. Debemos asegurarnos de ejecutar **commit()** o los comandos no se ejecutarán y los datos se perderán.

```
16 con.commit()
17 con.close()
```

Estos datos son persistentes y se pueden volver a cargar posteriormente.

**ATENCIÓN!!!!:** No debes llamar a tus archivos .py con el mismo nombre de las librerías que estás probando. Python intentará encontrar los métodos invocados en ese archivo y generará un error. **Esto es válido para cualquier import con el que estemos trabajando.**

## Ahorrando líneas

Si nos encontramos en la situación en la que ya existen datos para nuestra base de datos, tal vez necesitemos hacer una carga inicial.

Podemos agregar varios registros en la misma instrucción, cargándolos previamente a una lista e invocar el método **executemany()**, enviando una consulta paramétrica a la base de datos.



```
13 lista = [  
14     ("2154489", "Sebastián", "Álvarez", 7.75),  
15     ("2154563", "Analía", "Rovere", 8.75),  
16     ("2154896", "Jaime", "Felice", 8),  
17     ("2154492", "Rosa", "Medina", 9)  
18 ]  
19 cur.executemany('INSERT INTO alumnos VALUES (?, ?, ?, ?)', lista)
```

Debemos recordar comentar las líneas en las que agregamos datos una vez ejecutados, para que el programa no intente volver a agregarlos en la siguiente ejecución del programa.

Podrás conseguir el código completo de este ejemplo en el archivo ejemplo\_sqlite3.py

### Herramientas adicionales (opcional)

**DB Browser for SQLite:** <https://sqlitebrowser.org/dl/>

Programa de interfaz gráfica sencillo y liviano para visualizar rápidamente una base de datos en SQLite.

## 2. GUI: Interfaces gráficas en Python

Una GUI (Graphical User Interface), o interfaz de usuario, es la parte de un programa que proporciona un entorno visual simple que facilita la comunicación entre el usuario y un programa. Nos va a permitir poder utilizar nuestros programas mas allá de un entorno de consola o líneas en intérprete Python.

Está formada por un conjunto de gráficos como ventanas, botones, menús, casillas de verificación, etc.

### Características

- Es lo primero que percibe el usuario
- Es la parte con la que el usuario está en contacto.
- Los usuarios dependen de la interfaz para poder utilizar las funcionalidades del programa.
- El centro del diseño de una GUI es el usuario.

## Tkinter

Tkinter es un marco de interfaz gráfica integrado a la biblioteca de lenguaje standard de Python (es puente entre Python y la librería TCL/TK). Es multiplataforma: los elementos se representan utilizando elementos nativos del cada uno de los sistemas operativos.

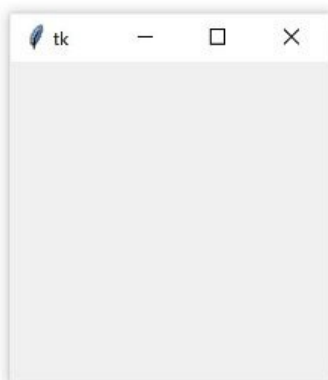
Podemos verificar la instalación de Tkinter ejecutando la siguiente línea:

```
python -m tkinter
```

Si bien el aspecto de las GUI creadas en Tkinter no se ven elegantes o modernas, Tkinter es muy liviano y relativamente fácil de usar en comparación con otros marcos. Esto lo convierte en una opción atractiva para empezar a crear aplicaciones GUI en Python rápidamente, que sean funcionales y multiplataforma.

## La ventana

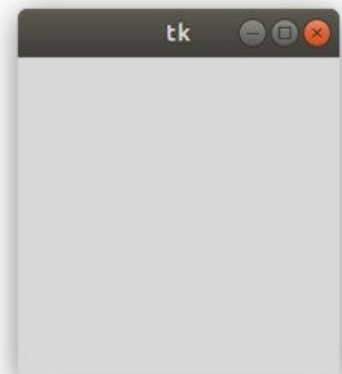
Es el elemento fundamental de una GUI. Es el contenedor en el que residen todos los demás elementos. Todos los elementos de la GUI contenidos dentro de la ventana (cuadros de texto, etiquetas, botones) se conocen como widgets.



(a) Windows



(b) macOS



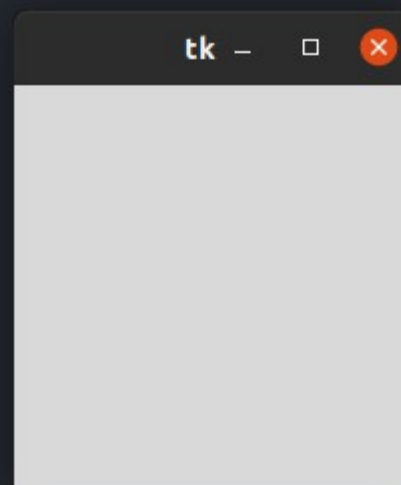
(c) Ubuntu

## Nuestra primera aplicación GUI

Crearemos una ventana que contiene un widget para aprender el procedimiento básico.



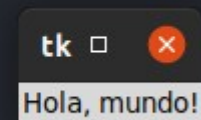
```
>>> import tkinter as tk  
>>> root = tk.Tk()  
>>> []
```



## Widgets

Los widgets son los elementos a través de los cuales los usuarios van a interactuar con el programa.

```
>>> root = tk.Tk()  
>>> hola = tk.Label(text='Hola, mundo!')  
>>> hola.pack()  
>>> root.mainloop()  
[]
```



- **tkinter.Label():** agrega texto a una ventana. En el ejemplo, creamos una etiqueta de texto y lo asignamos a la variable *hola*.
- **pack():** agrega el widget a la ventana. La ventana queda a la escala más pequeña posible que permita contener al widget.
- **mainloop():** indica a Python que ejecute en loop los eventos de Tkinter. Queda a la escucha de eventos (clicks, pulsaciones de teclas, etc.) y bloquea la ejecución de cualquier código hasta que se cierre la ventana que lo invoca.

## Otros tipos de widgets

- Button
- Canvas
- Checkbutton
- Entry
- Frame
- Label
- LabelFrame
- Listbox
- Menu
- Menubutton
- Message
- OptionMenu
- PanedWindow
- Radiobutton
- Scale
- Scrollbar
- Spinbox
- Text

Documentación Tkinter:

<https://docs.python.org/es/3/library/tk.html>

<https://guia-tkinter.readthedocs.io/es/develop/chapters/6-widgets/6.1-Intro.html>

## Entendiendo el código - Tkinter

En este capítulo vamos a interpretar las líneas que utilizamos para crear nuestra primera interfaz gráfica.

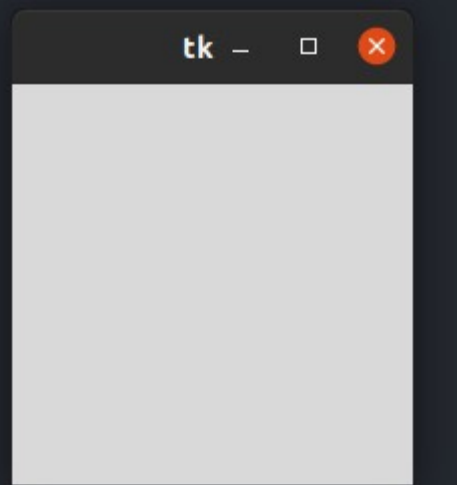
En la primera línea, importamos una librería: **tkinter**, y le asignamos un alias (**tk**), para poder referirnos a ella con un nombre más corto.

Recordamos que una librería en uno o varios archivos de código que nos proporcionan funcionalidades. Nos permiten invocar métodos, sin tener que preocuparnos en programarlos.

Al invocar esta librería, ya tenemos disponibles todas sus clases para poder crear objetos a partir de ellas.



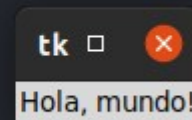
```
>>> import tkinter as tk
>>> root = tk.Tk()
>>> []
```



En la segunda línea, creamos una variable, y en ella almacenamos un objeto de la clase Tk (una ventana principal). Cada vez que creamos un objeto de una clase de la librería tkinter, debemos recordar agregar el alias previo al nombre de la clase, para indicar que busque la *plantilla* de nuestro nuevo objeto en aquella librería.

Luego creamos una etiqueta de texto (un objeto de tipo **Label**) y lo almacenamos en la variable **hola**. Como podemos ver, la clase admite que le enviemos el valor del atributo **text** como argumento.

```
>>> root = tk.Tk()
>>> hola = tk.Label(text='Hola, mundo!')
>>> hola.pack()
>>> root.mainloop()
[]
```



Para ubicar nuestro objeto **hola (Label)** en la ventana que creamos previamente, utilizamos el método **pack()** de la clase **Label**, invocándola por medio de nuestro objeto **hola**, seguido de punto.

La última instrucción que debemos agregar a nuestras GUI, es el método **mainloop()** de la clase **TK** (para ventanas principales). Como nosotros ya tenemos nuestra ventana principal almacenada en la variable **root**, debemos invocar al método a través de ella.

Este método mantiene nuestra ventana abierta y atenta a los eventos que ocurran durante la interacción del usuario, hasta que éste decida cerrarla o salir del programa.

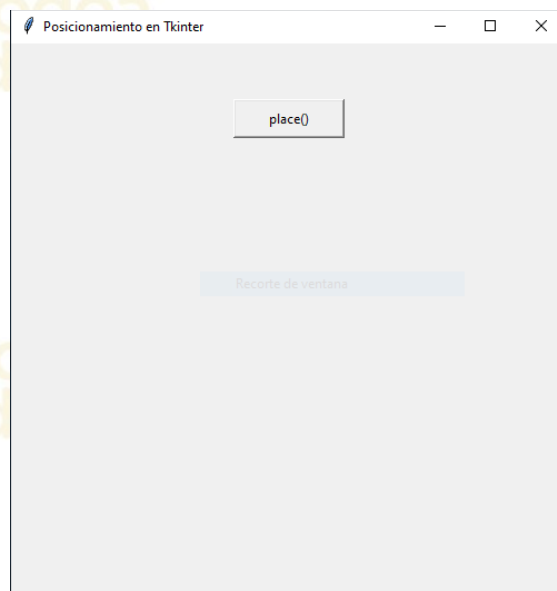
## Posicionando elementos en tkinter

Tkinter ofrece tres métodos para posicionar los widgets dentro de una ventana.

### Posición absoluta: place()

Permite ubicar los widgets indicando la posición x e y respecto de su elemento padre.

```
1  from tkinter import *
2  root = Tk()
3  root.title('Posicionamiento en Tkinter')
4
5  mi_frame = Frame(root)
6  mi_frame.config(width=500, height=500)
7  mi_frame.pack()
8
9  botonplace = Button(mi_frame, text='place()')
10 botonplace.place(x = 200, y = 50, width = 100, height = 35)
11
12 root.mainloop()
```



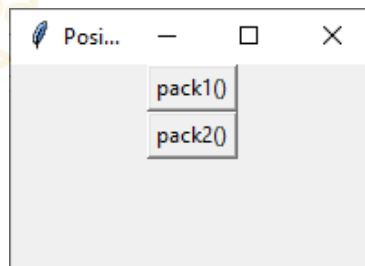


## Posicionamiento relativo: pack()

Especifica si el elemento se ubica abajo, arriba, a la derecha o izquierda respecto de algún otro elemento o la ventana principal.

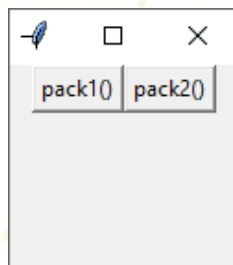
Si no se especifica ningún argumento, ubica los elementos uno encima del otro.

```
8
9  botonpack1 = Button(mi_frame, text='pack1()')
10 botonpack2 = Button(mi_frame, text='pack2()')
11 botonpack1.pack()
12 botonpack2.pack()
13
```



La propiedad que controla la posición relativa de los elementos es `side`, que puede tomar los valores `TOP`, `BOTTOM`, `LEFT` o `RIGHT`.

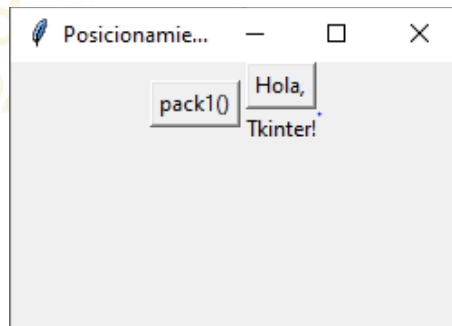
```
9  botonpack1 = Button(mi_frame, text='pack1()')
10 botonpack2 = Button(mi_frame, text='pack2()')
11 botonpack1.pack(side=LEFT)
12 botonpack2.pack()
```



La función pack también admite los parámetros after y before, que permiten controlar el orden en el que se ubican los elementos de la ventana.

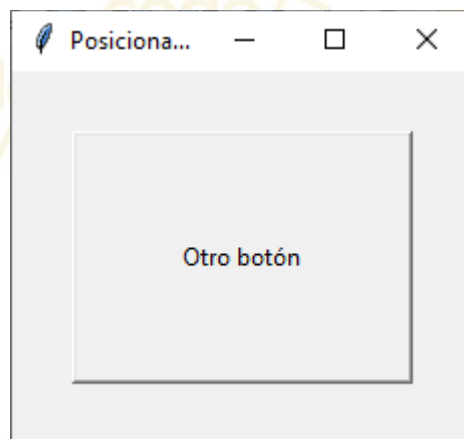
Tanto before como after, aceptan como valor cualquier widget para tomar como referencia.

```
9 botonpack1 = Button(mi_frame, text='pack1()')
10 botonpack1.pack(side=LEFT)
11 labelpack2 = Label(mi_frame, text = 'Tkinter!')
12 labelpack2.pack()
13 botonpack3 = Button(mi_frame, text='Hola,')
14 botonpack3.pack(before=labelpack2)
```



Otras propiedades que podemos incluir son padx, pady, ipadx e ipady. Especifican los márgenes externos e internos de un elemento.

```
9 boton4 = Button(mi_frame, text='Otro botón')
10 boton4.pack(padx=30, pady=30, ipadx=50, ipady=50)
```

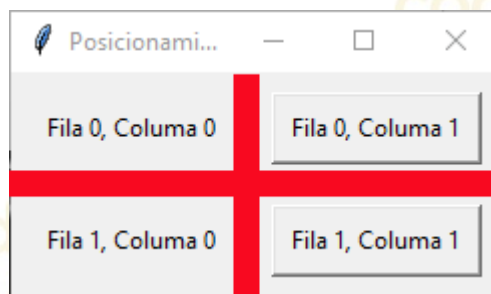




## Posicionamiento en forma de grilla: grid()

Consiste en dividir la ventana principal en filas (rows) y columnas (columns), formando celdas para ubicar los elementos

```
5  label1 = Label(root, text='Fila 0, Columa 0')
6  label1.grid(row=0, column=0, padx=10, pady=10, ipadx=5, ipady=5)
7
8  boton1 = Button(root, text='Fila 0, Columa 1')
9  boton1.grid(row=0, column=1, padx=10, pady=10, ipadx=5, ipady=5)
10
11 label2 = Label(root, text='Fila 1, Columa 0')
12 label2.grid(row=1, column=0, padx=10, pady=10, ipadx=5, ipady=5)
13
14 boton2 = Button(root, text='Fila 1, Columa 1')
15 boton2.grid(row=1, column=1, padx=10, pady=10, ipadx=5, ipady=5)
16
```



Podemos justificar nuestro texto con la opción 'sticky', siendo:

- sticky = 'w' alineado a la izquierda
- sticky = 'e' alineado a la derecha

*No es conveniente mezclar los métodos de posicionamiento.*

## Modificando elementos en tkinter

Podemos cambiar la apariencia y comportamiento de nuestros widgets. Veremos a continuación algunos de los widgets y sus opciones más comunes.

### Opciones comunes

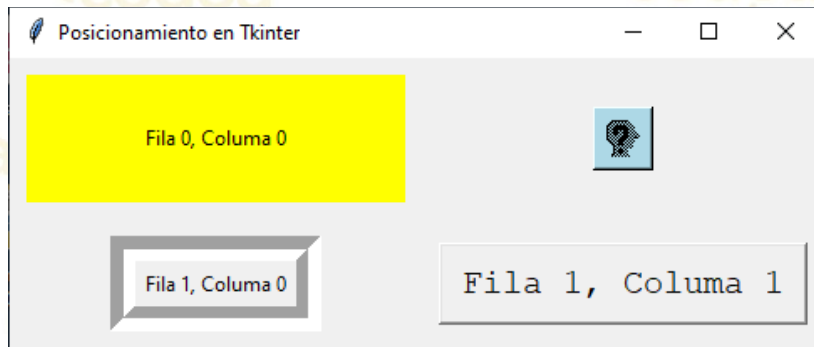
La mayoría de nuestros widgets permiten editar su apariencia.

- bg: color de fondo
- bd: ancho del borde en pixeles (por defecto 2px)
- relief: especifica el tipo de borde. Algunos de los valores son SUNKEN, RAISED, GROOVE y RIDGE
- font: tipo de fuente
- fg: color de la fuente
- 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
- cursor: permite cambiar el tipo de cursor cuando se posiciona sobre el elemento.

Opciones disponibles: <https://tkdocs.com/shipman/cursors.html>

```
5  label1 = Label(root, text='Fila 0, Columna 0', justify=LEFT, bg='yellow', width=30, height=4)
6  label1.grid(row=0, column=0, padx=10, pady=10, ipadx=5, ipady=5)
7
8  boton1 = Button(root, bitmap='questhead',bg='lightblue', activebackground='black')
9  boton1.grid(row=0, column=1, padx=10, pady=10, ipadx=5, ipady=5)
10
11 label2 = Label(root, text='Fila 1, Columna 0', bd=15, relief=GROOVE)
12 label2.grid(row=1, column=0, padx=10, pady=10, ipadx=5, ipady=5)
13
14 boton2 = Button(root, text='Fila 1, Columna 1', font=('Courier New', 15), activebackground='magenta')
15 boton2.grid(row=1, column=1, padx=10, pady=10, ipadx=5, ipady=5)
16 boton2.config(cursor="spider")
```





Entre el material de esta unidad podrás encontrar una tabla completa con los colores que maneja tkinter. Archivo: colores\_tkinter.png

Colores disponibles en Tkinter											
snow	deep sky blue	gold	seashell3	SlateBlue2	LightBlue3	SpringGreen2	DarkGoldenrod1	brown4	pink3	purple1	gray26
ghost white	sky blue	light goldenrod	seashell4	SlateBlue3	LightBlue4	SpringGreen3	DarkGoldenrod2	salmon1	pink4	purple2	gray27
white smoke	light sky blue	goldenrod	AntiqueWhite1	SlateBlue4	LightCyan2	SpringGreen4	DarkGoldenrod3	salmon2	LightPink1	purple3	gray28
gainsboro	steel blue	dark goldenrod	AntiqueWhite2	RoyalBlue1	LightCyan3	green2	DarkGoldenrod4	salmon3	LightPink2	purple4	gray29
floral white	light steel blue	rosy brown	AntiqueWhite3	RoyalBlue2	LightCyan4	green3	RosyBrown1	salmon4	LightPink3	MediumPurple1	gray30
old lace	light blue	indian red	AntiqueWhite4	RoyalBlue3	PaleTurquoise1	green4	RosyBrown2	LightSalmon2	LightPink4	MediumPurple2	gray31
linen	powder blue	saddle brown	bisque2	RoyalBlue4	PaleTurquoise2	chartreuse2	RosyBrown3	LightSalmon3	PaleVioletRed1	MediumPurple3	gray32
antique white	pale turquoise	sandy brown	bisque3	blue2	PaleTurquoise3	chartreuse3	RosyBrown4	LightSalmon4	PaleVioletRed2	MediumPurple4	gray33
papaya whip	dark turquoise	dark salmon	bisque4	blue4	PaleTurquoise4	chartreuse4	IndianRed1	orange2	PaleVioletRed3	thistle1	gray34
blanched almond	medium turquoise	salmon	PeachPuff3	DodgerBlue2	CadetBlue1	OliveDrab1	sienna2	orange3	PaleVioletRed4	thistle2	gray35
bisque	turquoise	light salmon	PeachPuff4	DodgerBlue3	CadetBlue2	OliveDrab2	IndianRed3	orange4	maroon1	thistle3	gray36
peach puff	cyan	orange	PeachPuff4	DodgerBlue4	CadetBlue3	OliveDrab4	IndianRed4	DarkOrange1	maroon2	thistle4	gray37
navajo white	light cyan	dark orange	NavajoWhite2	SteelBlue1	CadetBlue4	DarkOliveGreen1	sienna1	DarkOrange2	maroon3		gray38
lemon chiffon	cadet blue	coral	NavajoWhite3	SteelBlue2	turquoise1	DarkOliveGreen2	sienna2	DarkOrange3	maroon4		gray39
mint cream	medium aquamarine	light coral	NavajoWhite4	SteelBlue3	turquoise2	DarkOliveGreen3	sienna3	DarkOrange4	VioletRed1		gray40
azure	aquamarine	tomato	LemonChiffon2	SteelBlue4	turquoise3	DarkOliveGreen4	sienna4	coral1	VioletRed2		gray41
alice blue	dark green	orange red	LemonChiffon3	DeepSkyBlue2	turquoise4	khaki1	burlywood1	coral2	VioletRed3		gray42
lavender	dark olive green	red	LemonChiffon4	DeepSkyBlue3	cyan2	khaki2	burlywood2	coral3	VioletRed4		gray43
lavender blush	dark sea green	hot pink	cornsilk2	DeepSkyBlue4	cyan3	khaki3	burlywood3	coral4	magenta2		gray44
misty rose	sea green	deep pink	cornsilk3	SkyBlue1	cyan4	khaki4	burlywood4	tomato2	magenta3		gray45
dark slate gray	medium sea green	pink	cornsilk4	SkyBlue2	DarkSlateGray1	LightGoldenrod1	wheat1	tomato3	magenta4		gray46
dim gray	light sea green	light pink	ivory2	SkyBlue3	DarkSlateGray2	LightGoldenrod2	wheat2	tomato4	orchid1		gray47
slate gray	pale green	pale violet red	ivory3	SkyBlue4	DarkSlateGray3	LightGoldenrod3	wheat3	OrangeRed2	orchid2		gray48
light slate gray	spring green	maroon	ivory4	LightSkyBlue1	DarkSlateGray4	LightGoldenrod4	wheat4	OrangeRed3	orchid3		gray49
gray	lawn green	medium violet red	honeydew2	LightSkyBlue2	aquamarine2	LightYellow2	tan1	OrangeRed4	orchid4		gray50
light gray	medium spring green	violet red	honeydew3	LightSkyBlue3	aquamarine4	LightYellow3	tan2	red2	plum1		gray51
midnight blue	green yellow	medium orchid	honeydew4	LightSkyBlue4	DarkSeaGreen1	LightYellow4	tan4	red3	plum2		gray52
navy	lime green	dark orchid	LavenderBlush2	SlateGray1	DarkSeaGreen2	yellow2	chocolate1	red4	plum3		gray53
cornflower blue	yellow green	dark violet	LavenderBlush3	SlateGray2	DarkSeaGreen3	yellow3	chocolate2	DeepPink2	plum4		gray54
dark slate blue	forest green	blue violet	LavenderBlush4	SlateGray3	DarkSeaGreen4	yellow4	chocolate3	DeepPink3	MediumOrchid1		gray55
slate blue	olive drab	purple	MistyRose2	SlateGray4	SeaGreen1	gold2	firebrick1	DeepPink4	MediumOrchid2		gray56
medium slate blue	dark khaki	medium purple	MistyRose3	LightSteelBlue1	SeaGreen2	gold3	firebrick2	HotPink1	MediumOrchid3		gray57
light slate blue	khaki	thistle	MistyRose4	LightSteelBlue2	SeaGreen3	gold4	firebrick3	HotPink2	MediumOrchid4		gray58
medium blue	pale goldenrod	snow2	azure2	LightSteelBlue3	PaleGreen1	goldenrod1	firebrick4	HotPink3	DarkOrchid1		gray59
royal blue	light goldenrod yellow	snow3	azure3	LightSteelBlue4	PaleGreen2	goldenrod2	brown1	HotPink4	DarkOrchid2		gray60
blue	light yellow	snow4	azure4	LightBlue1	PaleGreen3	goldenrod3	brown2	pink1	DarkOrchid3		gray61
dodger blue	yellow	seashell2	SlateBlue1	LightBlue2	PaleGreen4	goldenrod4	brown3	pink2	DarkOrchid4		gray62



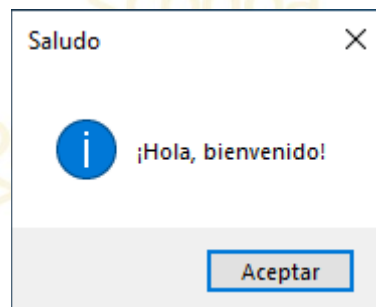
## Widgets I

### messagebox

Abre una ventana con un mensaje tipo modal, que nos proporciona un set de opciones según el tipo de mensaje que estamos presentando:

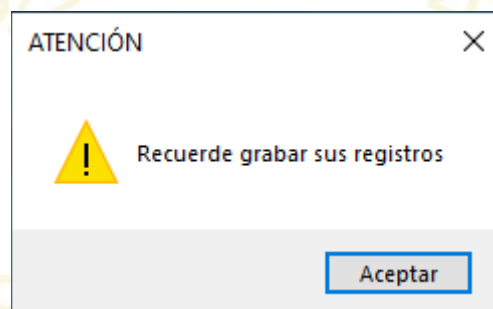
#### Ventanas de información

- i. `showinfo: tkinter.messagebox.showinfo(title=None, message=None, **options)`



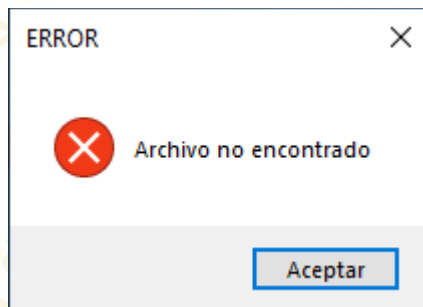
#### Ventanas de advertencia

- ii. `showwarning: tkinter.messagebox.showwarning(title=None, message=None, **options)`



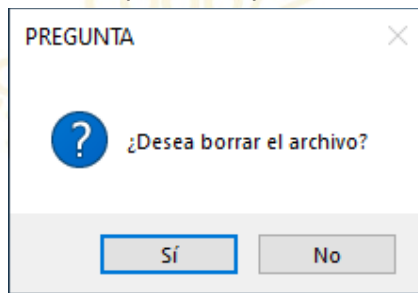
`showerror: tkinter.messagebox.showerror(title=None, message=None, **options)`



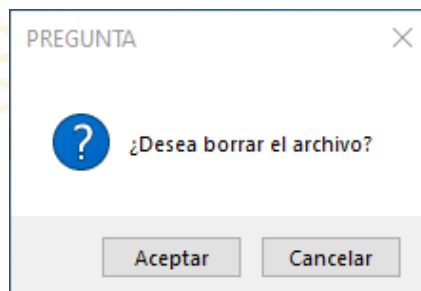


## Ventanas de opción

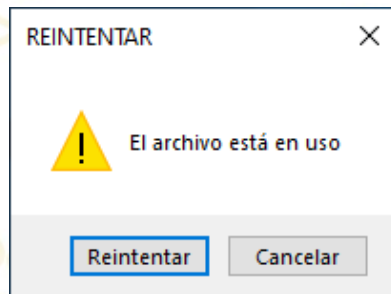
`askquestion: tkinter.messagebox.askquestion(title=None, message=None, **options)`



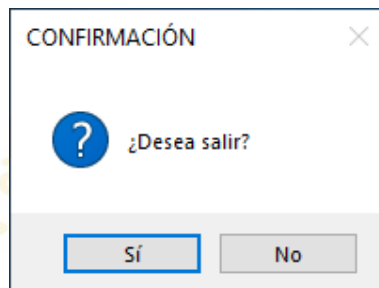
`askokcancel: tkinter.messagebox.askokcancel(title=None, message=None, **options)`



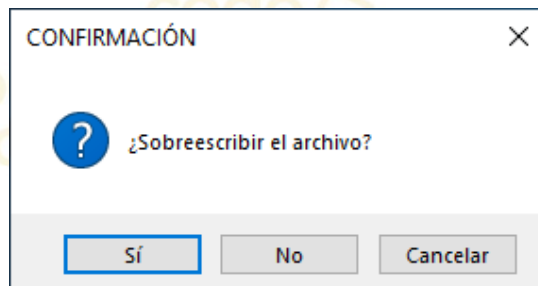
`askretrycancel: tkinter.messagebox.askretrycancel(title=None, message=None, **options)`



`askyesno: tkinter.messagebox.askyesno(title=None, message=None, **options)`



`askyesnocancel: tkinter.messagebox.askyesnocancel(title=None, message=None, **options)`



## Button

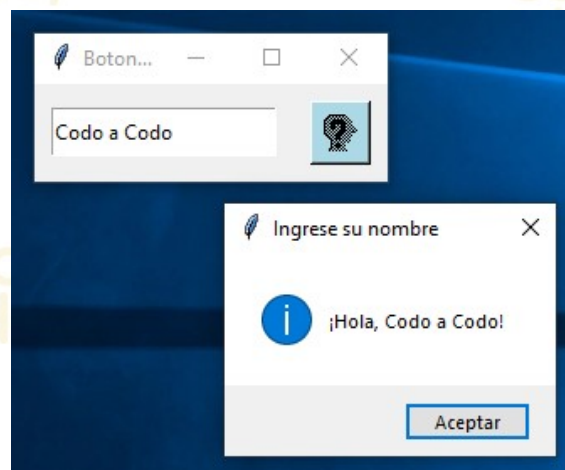
Las opciones que podemos agregar a las anteriores para modificar el comportamiento de nuestros botones son:

- `activebackground`: color de fondo cuando está encima el cursor
- `activeforeground`: idem anterior para color de la fuente.
- `command`: función o método a llamar cuando se clickea el botón



## Agregando funciones a los botones

```
1 from tkinter import *
2 from tkinter import messagebox
3 root = Tk()
4 root.title('Botones y funciones')
5
6 nombre = StringVar()
7
8 def saludar():
9     messagebox.showinfo(message=f'¡Hola, {nom.get()}!', title='Ingrese su nombre')
10    boton1.config(state=DISABLED)
11
12 nom = Entry(root, textvariable = nombre)
13 nom.grid(row=0, column=0, padx=10, pady=10, ipadx=5, ipady=5)
14 boton1 = Button(root, bitmap='questhead', bg='lightblue', activebackground='black', command=saludar)
15 boton1.grid(row=0, column=1, padx=10, pady=10, ipadx=5, ipady=5)
16
17 root.mainloop()
18
```



## Entry

Se utiliza para solicitar información al usuario.

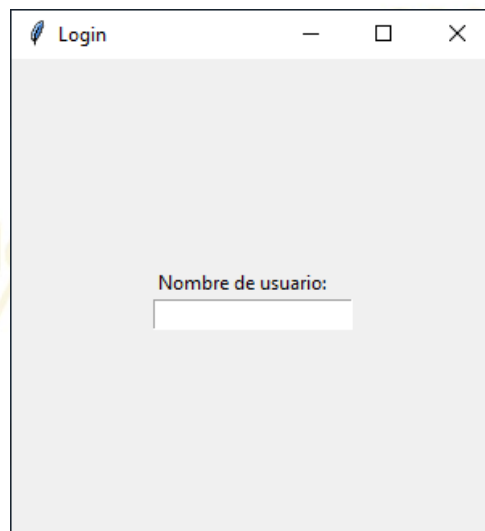
Sintaxis: `entry_name = tkinter.Entry(container, **options)`

Es conveniente utilizarlo junto a una variable que almacene su contenido.

# <codoa codo/>

```
1  from tkinter import *
2  from tkinter import messagebox
3
4  root = Tk()
5  root.title('Login')
6  root.geometry('300x300')
7  root.config(padx=20, pady=20)
8
9  mi_frame = Frame(root)
10 mi_frame.pack(expand=TRUE)
11
12 username = StringVar()
13
14 user_label = Label(mi_frame, text='Nombre de usuario:')
15 user_label.grid(row=1, column=0, sticky="w", padx=10)
16
17 user_entry = Entry(mi_frame, textvariable = username)
18 user_entry.grid(row=2, column=0, padx=10)
19
20 root.mainloop()
```

En este ejemplo, creamos una instancia de una variable tipo cadena, que almacenaremos en la variable 'user name'. Luego, signamos esta variable al widget Entry. Más adelante podremos utilizar un método para obtener el dato ingresado.



Agencia de  
Aprendizaje  
a lo largo  
de la vida



Los tipos de variable que tenemos disponibles son:

- 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 de tipo booleana

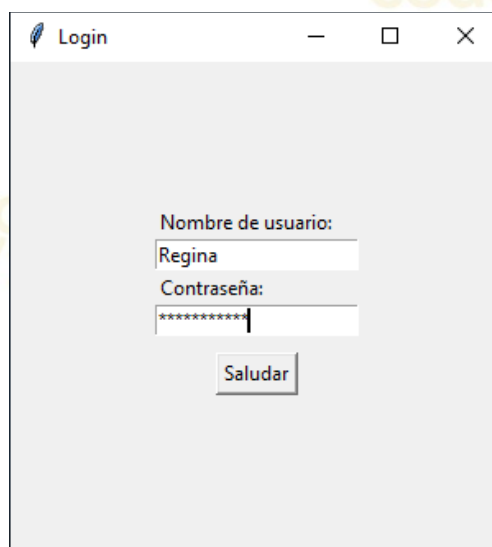
## Métodos:

- `entry_name.get()` : para obtener el contenido actual del campo Entry
- `entry_name.focus()` : podemos seleccionar el campo en el que queremos que se haga foco apenas aparece la ventana.
- `entry_name.set()` : permite escribir un texto por defecto en el campo Entry.

## Campos password

Con la opción 'show', podemos seleccionar un carácter que se muestre en lugar del texto que se tepea en el campo de entrada.

```
20 password = StringVar()
21
22 passw_label = Label(mi_frame, text='Contraseña:')
23 passw_label.grid(row=3, column=0, sticky="w", padx=10)
24
25 passw_entry = Entry(mi_frame, textvariable = password, show='*')
26 passw_entry.grid(row=4, column=0, padx=10)
```



Login

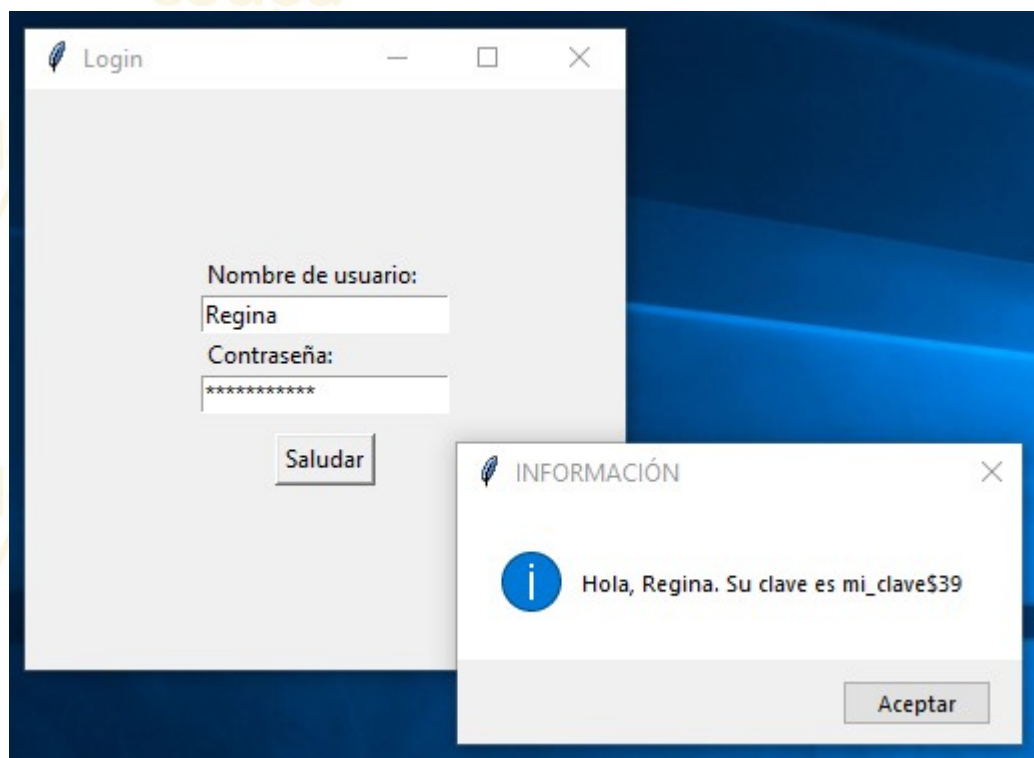
Nombre de usuario:  
Regina

Contraseña:  
\*\*\*\*\*

Saludar

## command

```
12 def saludo_button():
13     msg = f'Hola, {user_entry.get()}. Su clave es {passw_entry.get()}'
14     messagebox.showinfo(title='INFORMACIÓN', message=msg)
15     user_entry.set('')
16     passw_entry.set('')
17
32
33 saludo_button = Button(mi_frame, text='Saludar', command=saludo_button)
34 saludo_button.grid(row=5, column=0, padx=10, pady=10)
35
```



El código de este ejemplo está disponible en el archivo `unidad5a.py`



## OptionMenu

Se utiliza para crear un menú desplegable. Permite mostrar varias opciones en menos espacios. Sólo puede seleccionarse una de las opciones.

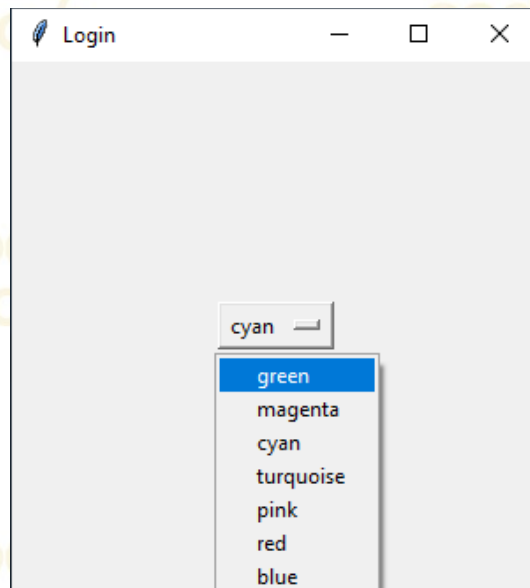
### Sintaxis

`OptionMenu(container, variable, value, *values, **kwargs)`

- `container`: es el contenedor donde ubicamos el `OptionMenu`. Puede ser la ventana principal, ventana secundaria o frame.
- `variable`: es la variable donde se almacena el valor que tome el `OptionMenu`. Esto es: la opción seleccionada.
- `value`: depende del tipo de variable que estamos manejando. Si es un `StringVar()` puede ser cualquier nombre o set de caracteres
- `*values`: es el nombre de la lista en la que ubicamos todas las opciones
- `**kwargs`: otras opciones de configuración

```
1  from tkinter import *
2  from tkinter import messagebox
3
4  root = Tk()
5  root.title('Login')
6  root.geometry('300x300')
7  root.config(padx=20, pady=20)
8
9  mi_frame = Frame(root)
10 mi_frame.pack(expand=TRUE)
11
12 def colorear(eleccion):
13     eleccion = mi_color.get()
14     print(eleccion)
15
16 colores = ['green', 'magenta', 'cyan', 'turquoise', 'pink', 'red', 'blue']
17
18 mi_color = StringVar()
19 mi_color.set(colores[2])
20
21 opt_menu = OptionMenu(mi_frame, mi_color, *colores, command=colorear)
22 opt_menu.pack()
23
24 root.mainloop()
```

# <codoa codo/>



```
on310/python.exe  
turquoise  
pink  
red  
green  
█
```

*Salida por terminal*

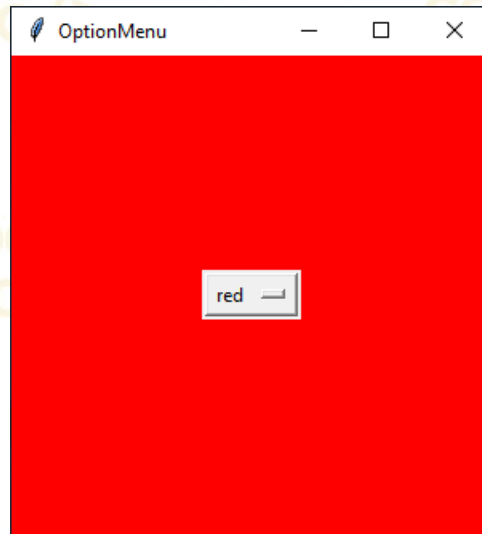
## Opciones

- Puede modificarse su aspecto con las opciones que mencionamos antes, en esta misma unidad: width/height, color, font, bg
- Contamos con los métodos set() para darle un valor inicial y get() para obtener el valor actual.

```
def colorear(eleccion):  
    eleccion = mi_color.get()  
    root.config(bg=eleccion)
```

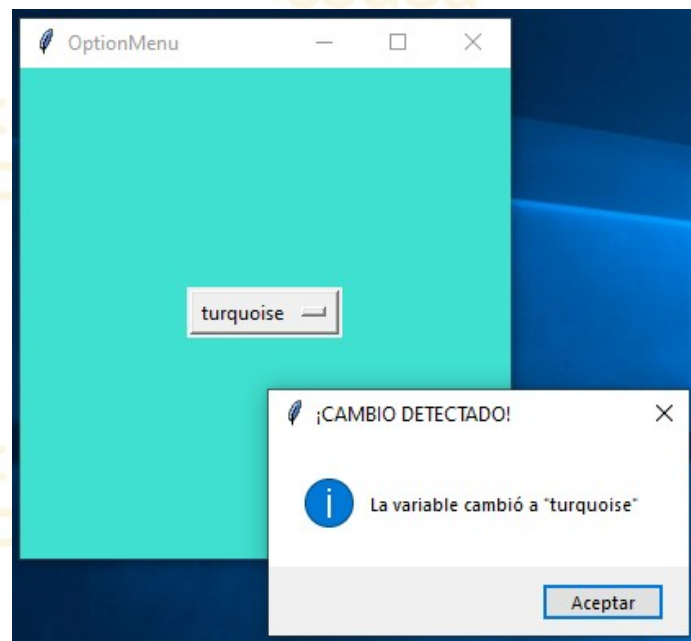
Agencia de  
Aprendizaje  
a lo largo  
de la vida

# <codoa codo/>



- change event / trace(): lleva un seguimiento de los cambios que ocurren en el widget OptionMenu

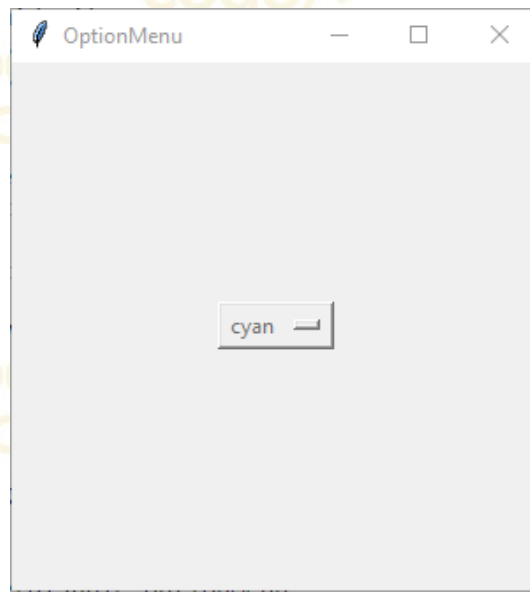
```
def check(*args):  
    messagebox.showinfo('¡CAMBIO DETECTADO!', f'La variable cambió a "{mi_color.get()}"')
```





- state: permite habilitar (normal) / deshabilitar (disabled) el menú.

```
opt_menu = OptionMenu(mi_frame,  
opt_menu.config(state=DISABLED)  
opt_menu.pack()
```



Encontrarás el código de estos ejemplos en el archivo `optionmenu.py`

## Imágenes

Finalmente, es posible agregar imágenes a nuestros proyectos, gracias a PhotoImage:

```
1  from tkinter import *
2
3  root = Tk()
4  root.title('Imagen')
5  root.geometry('300x300')
6  root.config(padx=20, pady=20)
7
8  mi_frame = Frame(root)
9  mi_frame.pack(expand=TRUE)
10
11  img = PhotoImage(file='ejemplos/py.png')
12  Label(mi_frame, image=img).pack(ipady=10)
13
14  mainloop()
```

