TKINTER PARA PYTHON

Curso de interfaces gráficas

Aprende a realizar aplicaciones de Python para escritorio
En el siguiente código QR tendrás acceso a los videotutoriales
publicados en YouTube del canal Programación Fácil.

Pere Manel Verdugo Zamora
pereverdugo@gmail.com

Capítulo 1: Ventana gráfica

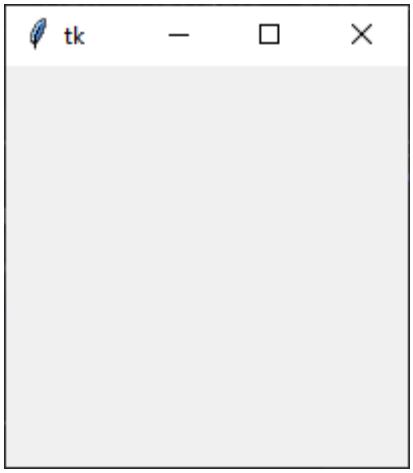
Tkinter es una librería que ya viene instalada en Python.

```
1    # Vamos a importar Tkinter
2    import tkinter as tk
3
4    root = tk.Tk()
5    root.mainloop()
```

En la línea 4 creamos un objeto llamado toot con la clase Tk() de Tkinter.

En la línea 5 el .mainloop() hace que se refresque la ventana constantemente, si no la ponemos cuando ejecutemos esta se cerrará después de ejecutarse.

Con esto tan simple ya tenemos la primera ventana:



Aparecerá con un tamaño y título por defecto.

```
1    # Vamos a importar Tkinter
2    from tkinter import *
3
4    root = Tk()
5    root.mainloop()
```

Si no queremos tener que hacer referencia al alias tk podemos importar de este modo, como verás en la línea 4 ya podemos omitir tk. y escribir directamente Tk().

Ahora vamos a agregar un texto por mediación de una etiqueta.

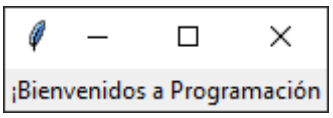
```
1    # Vamos a importar Tkinter
2    from tkinter import *
3
```

```
4    root = Tk()
5    etiqueta = Label(root, text="¡Bienvenidos a Programación")
6    etiqueta.pack()
7    root.mainloop()
```

En la línea 5 definimos un objeto llamado etiqueta, que será de la clase tkinter el método Label, como parámetros donde tiene que ir root y el texto que tiene que tener.

En la línea 6 le estamos diciendo que se ajuste a la ventana.

Este será el resultado:



Capítulo 2: ¿Qué son los widgets? – El widget Frame() y el método pack()

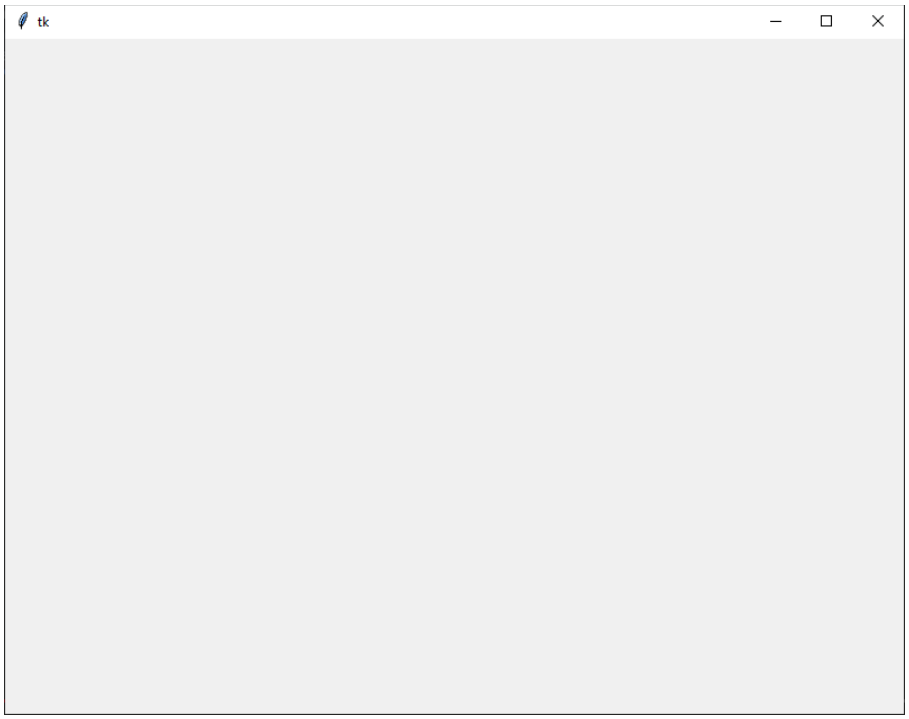
```
1   from tkinter import *
2
3   root = Tk()
4   etiqueta = Label(root, text="¡Bienvenidos a Programación")
5   marco_principal = Frame()
6   marco_principal.pack()
7   root.mainloop()
```

Cuando ejecutemos este será el resultado:

Vamos a personalizar el tamaño.

```
1   from tkinter import *
2
3   root = Tk()
4   etiqueta = Label(root, text="¡Bienvenidos a Programación")
5   marco_principal = Frame()
6   marco_principal.pack()
7   marco_principal.config(width="800", height="600")
8   root.mainloop()
```

Vamos a ejecutar:

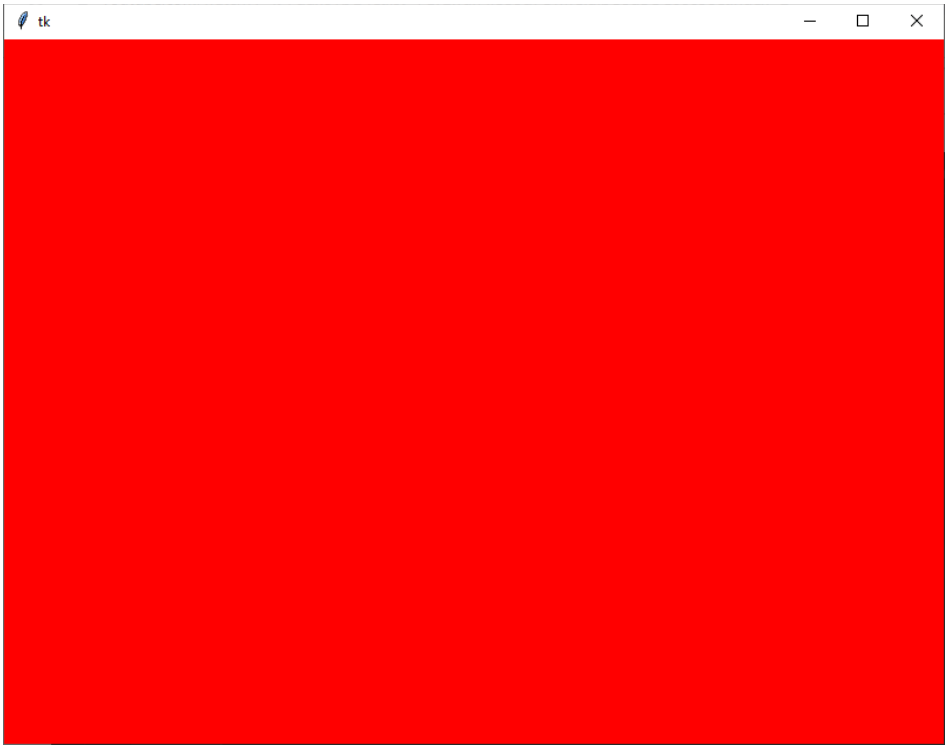


Al dar unas dimensiones al marco la ventana se redimensiona para que pueda entrar el marco.

Vamos a poner un fondo de color rojo.

```
1   from tkinter import *
2
3   root = Tk()
4   etiqueta = Label(root, text="¡Bienvenidos a Programación")
5   marco_principal = Frame()
6   marco_principal.pack()
7   marco_principal.config(width="800", height="600")
8   marco_principal.config(bg="red")
9   root.mainloop()
```

Vamos a ejecutar:

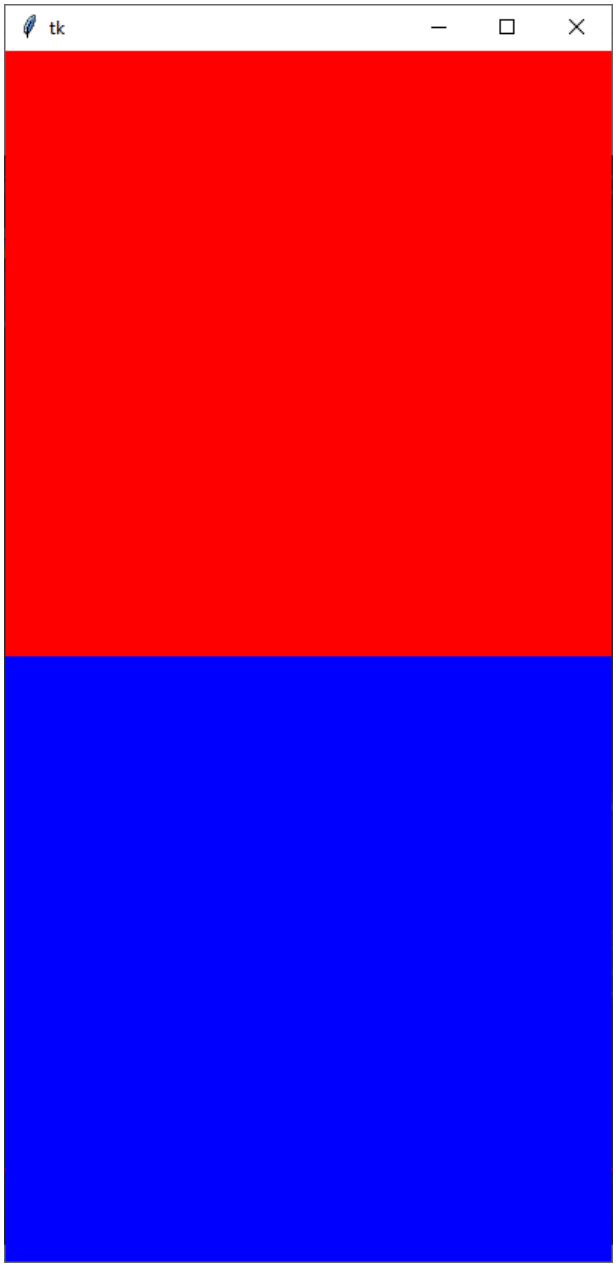


El widget Frame()

```
1   from tkinter import *
2   root = Tk()
3
4   marco_principal = Frame()
5   marco_principal.pack()
6
7   marco_principal2 = Frame()
8   marco_principal2.pack()
```

```
 9
10    marco_principal.config(width="400", height="400")
11    marco_principal.config(bg="red")
12
13    marco_principal2.config(width="400", height="400")
14    marco_principal2.config(bg="blue")
15
16    root.mainloop()
```

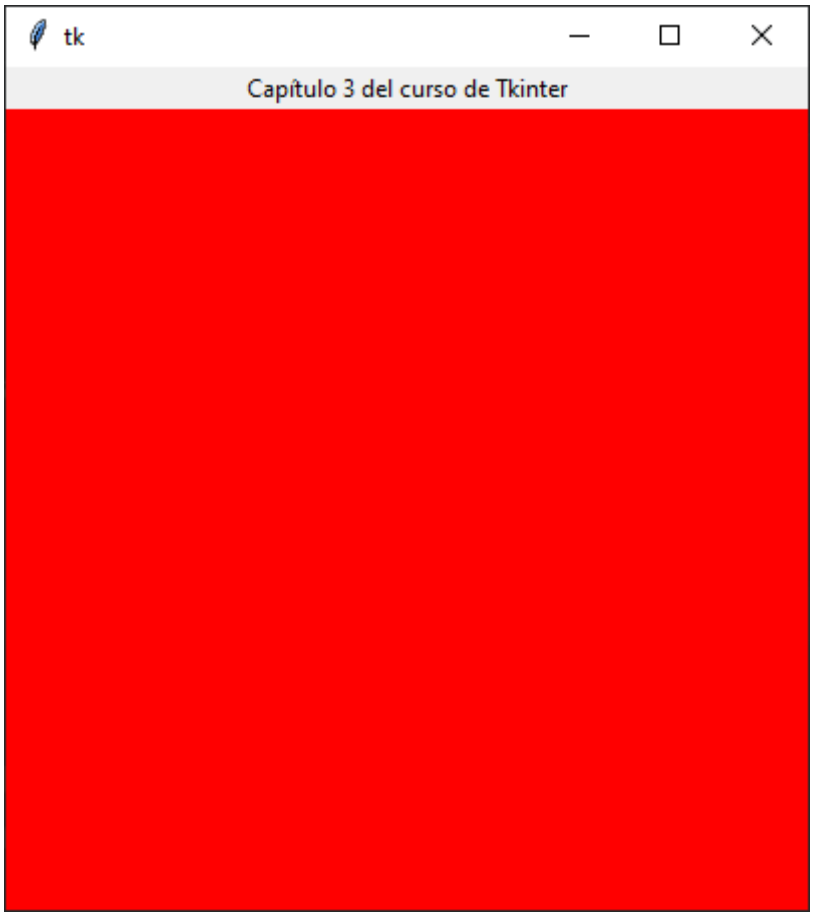
Este será el resultado:



Capítulo 3: El método grid()

```
1   from tkinter import *
2   root = Tk()
3
4   marco_principal = Frame()
5   texto = Label(root, text="Capítulo 3 del curso de Tkinter")
6
7   texto.pack()
8   marco_principal.pack()
9
10  marco_principal.config(width="400", height="400")
11  marco_principal.config(bg="red")
12
13  root.mainloop()
```

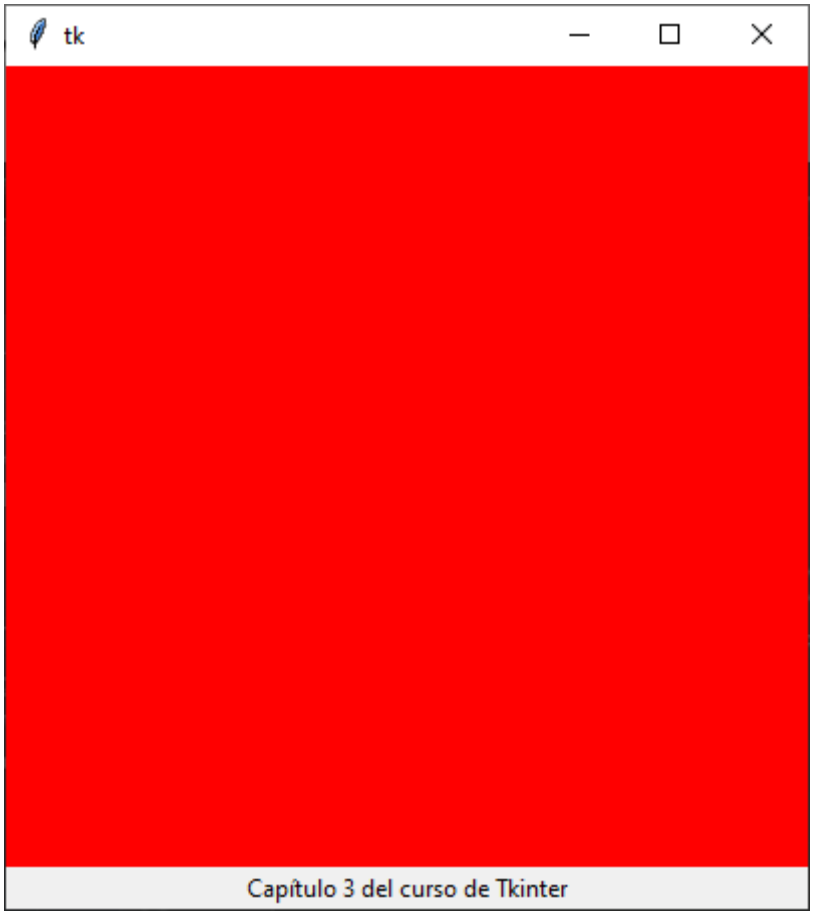
En el momento de utilizar el método pack() el orden es importante.



Vamos a cambiar el orden.

```
1   from tkinter import *
2   root = Tk()
3
4   marco_principal = Frame()
5   texto = Label(root, text="Capítulo 3 del curso de Tkinter")
6
7   marco_principal.pack()
8   texto.pack()
9
10  marco_principal.config(width="400", height="400")
11  marco_principal.config(bg="red")
12
13  root.mainloop()
```

Este será el resultado:

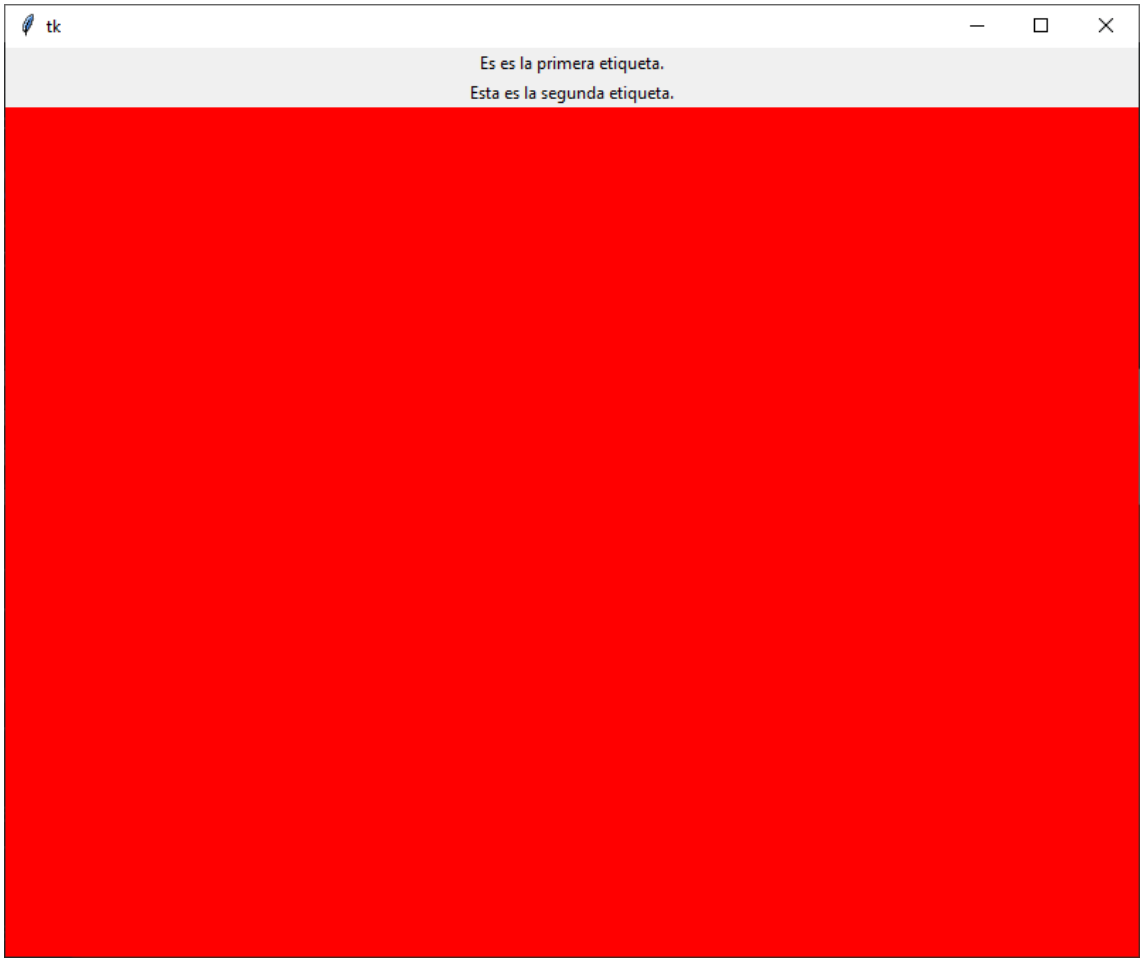


Otro ejemplo:

```
1   from tkinter import *
2   root = Tk()
3
4   # Etiqueta (Muestra un texto)
5
```

```
6   etiqueta = Label(root, text="Es es la primera etiqueta.")
7   etiqueta2 = Label(root, text="Esta es la segunda etiqueta.")
8   etiqueta.pack()
9   etiqueta2.pack()
10
11  # Creamos un marco
12  marco_principal = Frame()
13
14  # Empaquetamos el marco
15  marco_principal.pack()
16
17  # Redimensionamos el marco.
18  marco_principal.config(width="800", height="600")
19
20  # Le damos color al marco
21  marco_principal.config(bg="red")
```

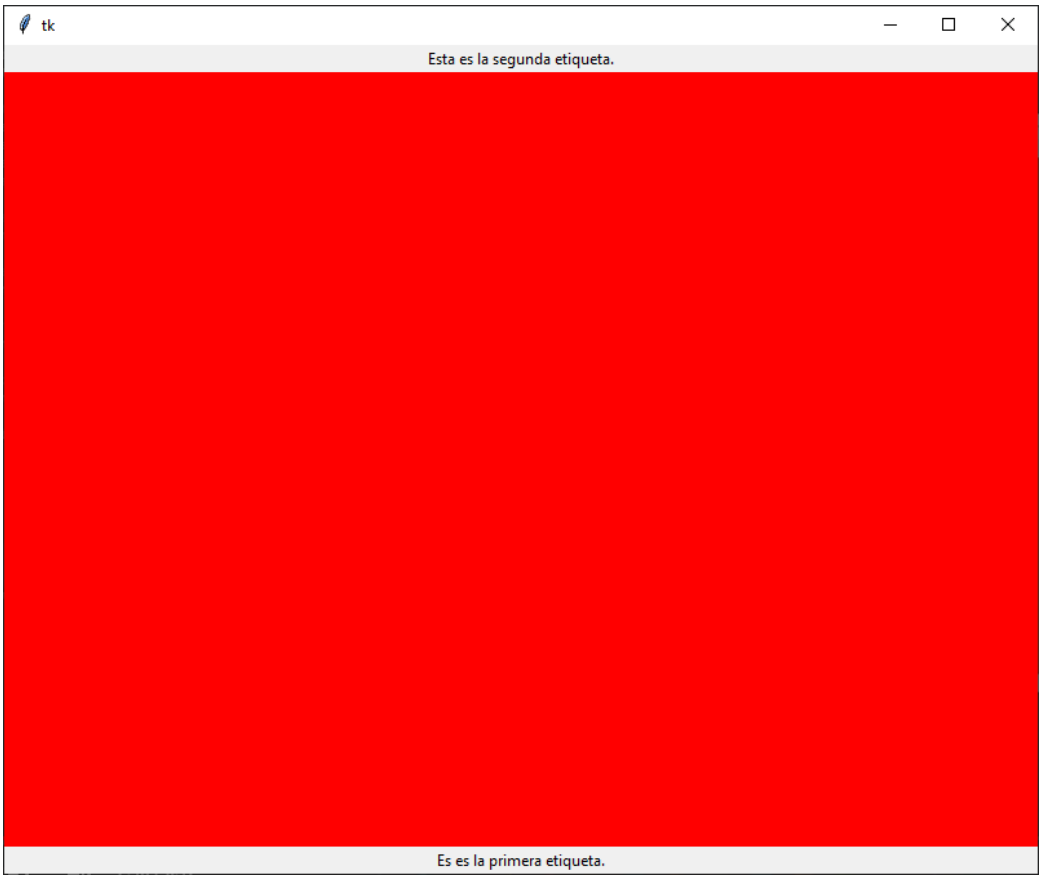
Este será el resultado:



Vamos a ver la diferencia con grid():

```
1   from tkinter import *
2   root = Tk()
3
4   # Etiqueta (Muestra un texto)
5
6   etiqueta = Label(root, text="Es es la primera etiqueta.")
7   etiqueta2 = Label(root, text="Esta es la segunda etiqueta.")
8   etiqueta.grid(row=2, column=0)
9   etiqueta2.grid(row=0, column=0)
10
11  # Creamos un marco
12  marco_principal = Frame()
13
14  # Empaquetamos el marco
15  marco_principal.grid(row=1, column=0 )
16
17  # Redimensionamos el marco.
18  marco_principal.config(width="800", height="600")
19
20  # Le damos color al marco
21  marco_principal.config(bg="red")
```

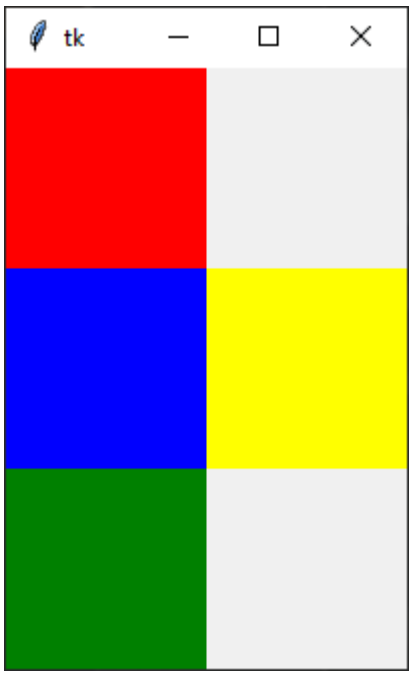
Este será el resultado:



Otro ejemplo:

```
from tkinter import *
root = Tk()

# Marco 1
marco_principal1 = Frame()
marco_principal1.grid(row=0, column=0)
marco_principal1.config(width="100", height="100")
marco_principal1.config(bg="red")

# Marco 2
marco_principal2 = Frame()
marco_principal2.grid(row=1, column=0)
marco_principal2.config(width="100", height="100")
marco_principal2.config(bg="blue")

# Marco 3
marco_principal3 = Frame()
marco_principal3.grid(row=1, column=1)
marco_principal3.config(width="100", height="100")
marco_principal3.config(bg="yellow")

# Marco 4
marco_principal4 = Frame()
marco_principal4.grid(row=2, column=0)
marco_principal4.config(width=100, height="100")
marco_principal4.config(bg="green")

root.mainloop()
```

Este será el resultado:



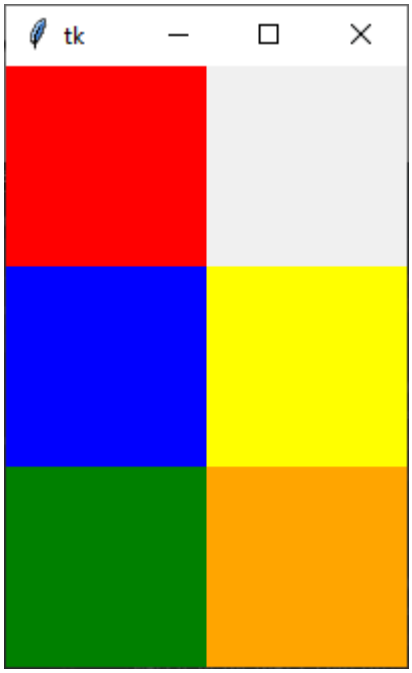
Queremos que el ultimo marco sea de color naranja.

```
from tkinter import *
root = Tk()

# Marco 1
marco_principal1 = Frame()
marco_principal1.grid(row=0, column=0)
marco_principal1.config(width="100", height="100")
marco_principal1.config(bg="red")

# Marco 2
marco_principal2 = Frame()
marco_principal2.grid(row=1, column=0)
marco_principal2.config(width="100", height="100")
marco_principal2.config(bg="blue")

# Marco 3
marco_principal3 = Frame()
marco_principal3.grid(row=1, column=1)
marco_principal3.config(width="100", height="100")
marco_principal3.config(bg="yellow")

# Marco 4
marco_principal4 = Frame()
marco_principal4.grid(row=2, column=0)
marco_principal4.config(width=100, height="100")
marco_principal4.config(bg="green")
```

```
# Marco 5
marco_principal5 = Frame()
marco_principal5.grid(row=2, column=1)
marco_principal5.config(width=100, height="100")
marco_principal5.config(bg="orange")
```

```
root.mainloop()
```

Este será el resultado:

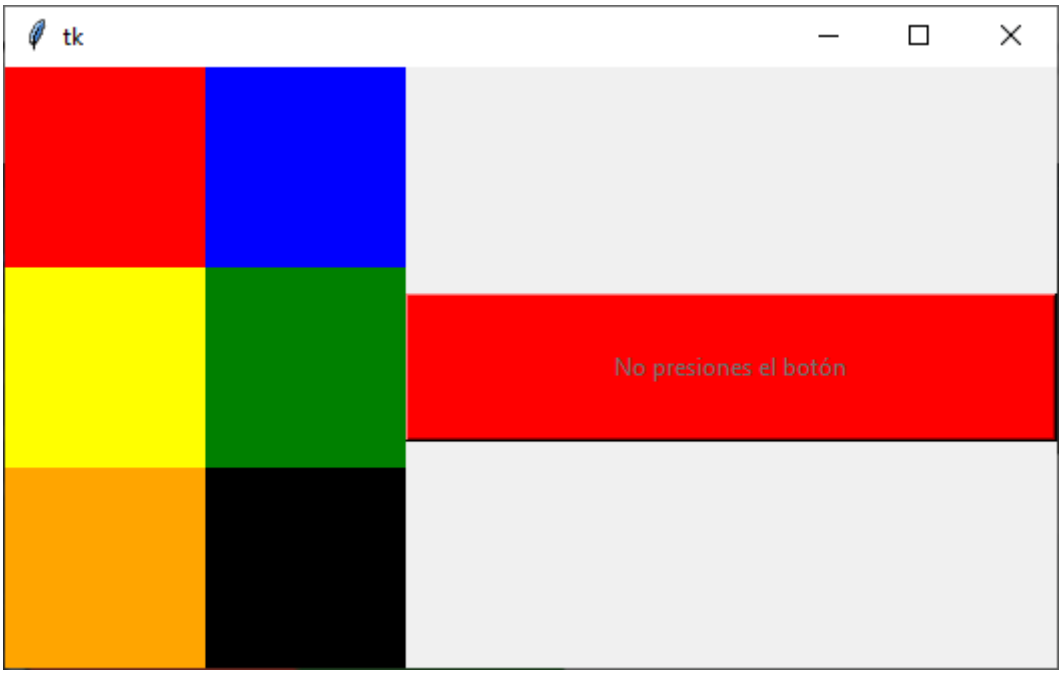


Capítulo 4: Relativas y el widget button()

```
from tkinter import *
root = Tk()

# Marco 1
marco_principal1 = Frame()
marco_principal1.grid(row=0, column=0)
marco_principal1.config(width="100", height="100")
marco_principal1.config(bg="red")

# Marco 2
marco_principal2 = Frame()
marco_principal2.grid(row=0, column=1)
marco_principal2.config(width="100", height="100")
marco_principal2.config(bg="blue")

# Marco 3
marco_principal3 = Frame()
marco_principal3.grid(row=1, column=0)
marco_principal3.config(width="100", height="100")
marco_principal3.config(bg="yellow")

# Marco 4
marco_principal4 = Frame()
marco_principal4.grid(row=1, column=1)
marco_principal4.config(width=100, height="100")
marco_principal4.config(bg="green")

# Marco 5
marco_principal5 = Frame()
marco_principal5.grid(row=2, column=0)
marco_principal5.config(width=100, height="100")
marco_principal5.config(bg="orange")

# Marco 6
marco_principal6 = Frame()
marco_principal6.grid(row=2, column=1)
marco_principal6.config(width=100, height="100")
marco_principal6.config(bg="black")

boton1 = Button(root, text="No presiones el botón", bg="red",padx=100,
pady=25, state=DISABLED).grid(row=1, column=2)

root.mainloop()
```

- Text → nos permite agregar un texto en el botón.
- bg → dar color al botón.

- padx → definir el ancho en pixeles.
- pady → definir el alto en pixeles.
- state → si le asignados DISABLE no permitirá presionarlo.
- grid → por mediación de row (fila) y columna (columna) controlamos su posición.

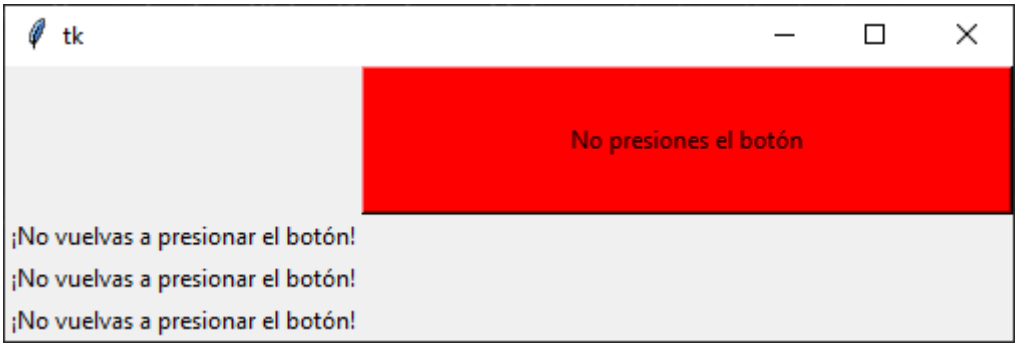


Capítulo 5: Llamar a funciones desde un botón

```
from tkinter import *
root = Tk()

def click_boton():
    texto = Label(root, text="¡No vuelvas a presionar el botón!").grid()

boton1 = Button(root, text="No presiones el botón", bg="red", padx=100,
pady=25, command=click_boton).grid(row=1, column=2)

root.mainloop()
```

Creamos una función llamada click_boton() que le asignamos a texto el objeto de tipo Label (etiqueta) con el texto "¡No vuelvas a presionar el botón!"

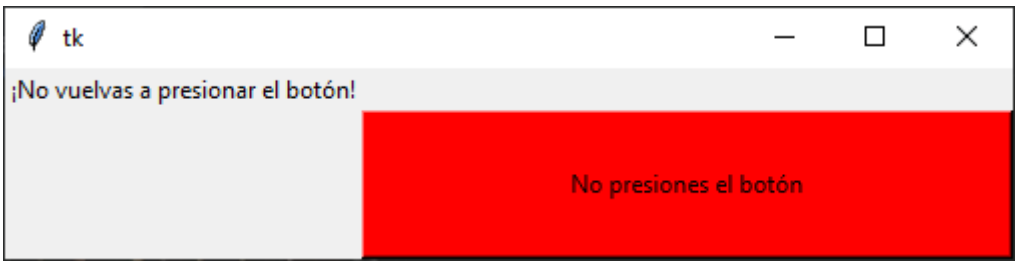
En el objeto boton1 de tipo Button con la propiedad command=click_boton le estamos diciendo que cuando presiones el botón ejecuta la función.



Hemos presionado al botón 3 veces.

```
boton1 = Button(root, text="No presiones el botón", bg="red", padx=100,
pady=25, command=click_boton()).grid(row=1, column=2)
```

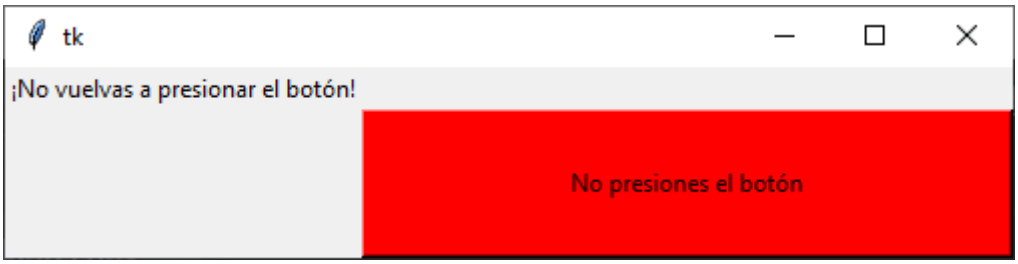
Si a la función le agregamos los paréntesis, cuando ejecutemos el programa esta será llamada una vez, pero cuando hagamos clic en el botón esta no responderá. Aparecerá una sola vez.



Si queremos que se ejecute una sola vez y cuando presionemos el botón.

```
def click_boton():
    texto = Label(root, text="¡No vuelvas a presionar el
botón!").grid(row=0, column=0)
```

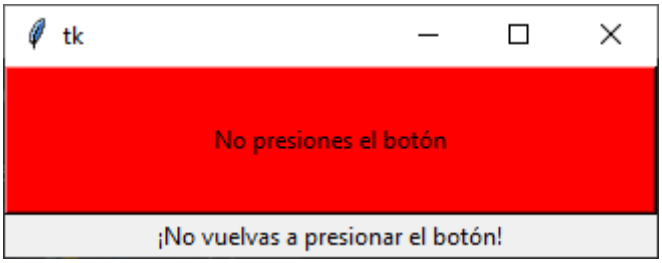
En el texto le daremos una posición determinada con el grid(row=0, columna=0).
Vamos a ejecutar y pulsar el botón varias veces.



¿Cómo haríamos si queremos que el texto parezca por debajo del botón?

```
def click_boton():
    texto = Label(root, text="¡No vuelvas a presionar el
botón!").grid(row=1, column=0)

boton1 = Button(root, text="No presiones el botón", bg="red", padx=100,
pady=25, command=click_boton).grid(row=0, column=0)
```

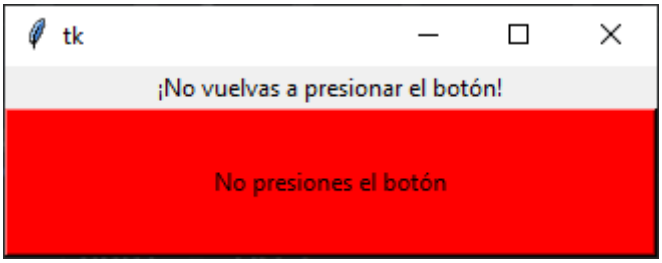
Modificando las columnas y filas del grid tanto en el objeto texto de la función click_boton(), como el objeto boton1 de tipo Button.
Cuando ejecutemos y hagamos un clic.



Queremos que el texto se muestre arriba.

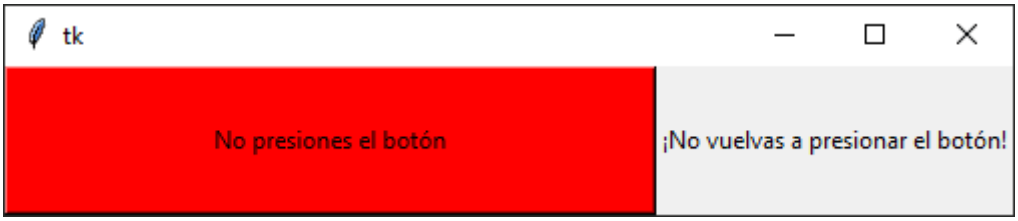
```
def click_boton():
    texto = Label(root, text="¡No vuelvas a presionar el
botón!").grid(row=0, column=0)

boton1 = Button(root, text="No presiones el botón", bg="red", padx=100,
pady=25, command=click_boton).grid(row=1, column=0)
```

Cambiaremos el valor de row (fila), este será el resultado:



Ejercicio práctico:

Ahora tú tienes que hacer quede de la siguiente forma:



Y ahora de la siguiente forma:

Capítulo 6: FORMULARIOS con el widget Entry() y CONTRASEÑAS protegidas

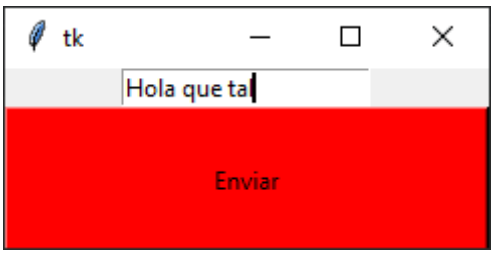
```
from tkinter import *
root = Tk()

entrada = Entry(root)
entrada.grid(row=0, column=0)

def click_boton():
    texto = Label(root, text="¡Se envio correctamente!").grid(row=0,
column=0)

boton1 = Button(root, text="Enviar", bg="red", padx=100, pady=25,
command=click_boton).grid(row=1, column=0)

root.mainloop()
```

Definimos un objeto de tipo Entry llamado entrada.

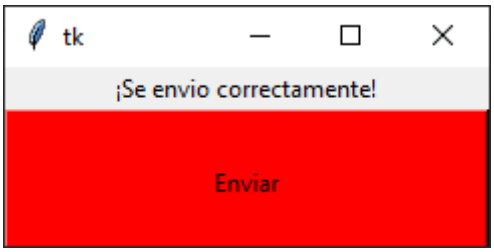
Lo posicionamos en la fila 0 columna 0.

En la función click:botón() el objeto texto de tipo Label que contiene el texto "¡Se envío correctamente! Y con las mismas coordenadas de fila 0 columna 0, al estar en las mismas coordenadas, un texto sustituirá al otro.

Vamos a ejecutar:



Escribimos un texto y presionamos el botón Enviar.

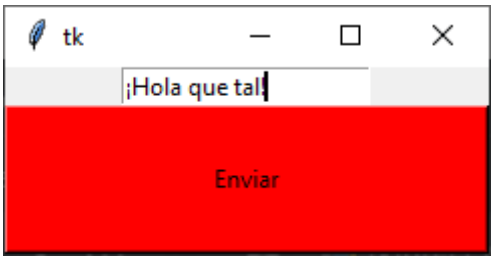


Si no quieres que se reemplace es cuestión de modificar las coordenadas.

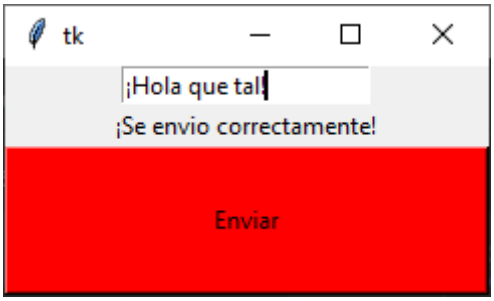
```
from tkinter import *
root = Tk()

entrada = Entry(root)
entrada.grid(row=0, column=0)

def click_boton():
    texto = Label(root, text="¡Se envio correctamente!").grid(row=1,
column=0)

boton1 = Button(root, text="Enviar", bg="red", padx=100, pady=25,
command=click_boton).grid(row=2, column=0)

root.mainloop()
```

Vamos a ejecutar:

Ponemos el texto.



Presionamos el botón.



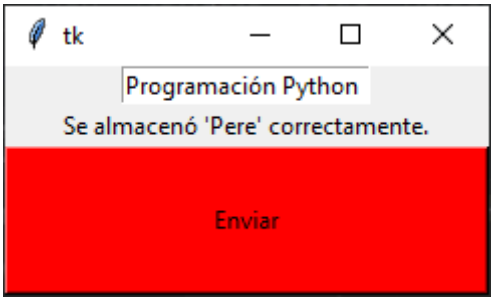
Queremos que nos muestre como mensaje el texto que hemos introducido.

```
from tkinter import *
root = Tk()

entrada = Entry(root)
entrada.grid(row=0, column=0)

def click_boton():
    texto = Label(root, text=f"Se almacenó '{entrada.get()}'
correctamente.").grid(row=1, column=0)
```

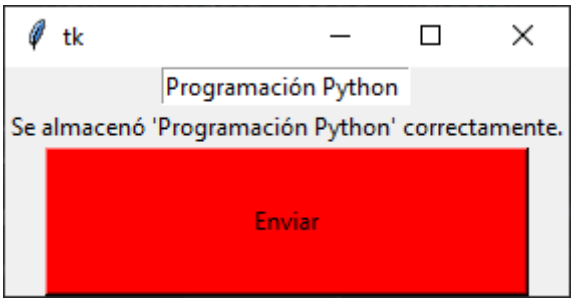
```
boton1 = Button(root, text="Enviar", bg="red", padx=100, pady=25,
command=click_boton).grid(row=2, column=0)

root.mainloop()
```

En la función click_boton() para mostrar en el mensaje el texto que hemos introducido anteriormente al objeto texto de tipo Label le asignamos un texto con formato donde recuperamos el valor con el nombre de la variable seguido de un punto y el método get, **entrada.get**.

Cuando ejecutemos e introduzcamos 'Programación Python'.



Presionamos el botón:



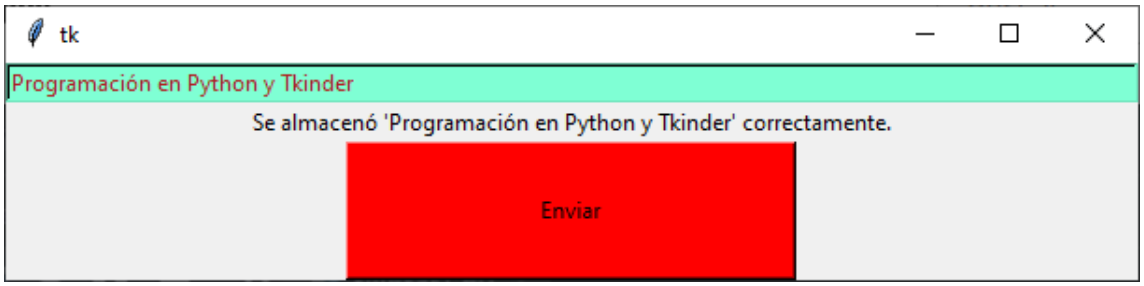
Ahora vamos a personalizarlo.

```
entrada = Entry(root, width=100, bg="aquamarine", fg="firebrick",
borderwidth=2)
entrada.insert(0, "Escriba aquí...")
```

En el objeto entrada con width controlamos el ancho, bg color de fondo, fg color de la fuente y borderwidth su borde.
Si además queremos que aparezca un texto por defecto que podremos sustituir escribiremos el nombre del objeto seguido del punto e insert con los parámetros en la posición y el texto.
Vamos a ejecutar:

Vamos a sustituir el texto por “Programación en Python y Tkinder.

Presionamos el botón:

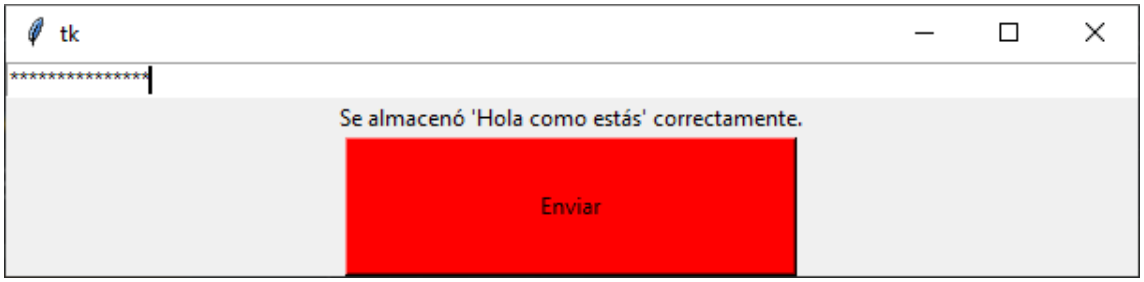


Ahora lo vamos a cambiar por:

```
entrada = Entry(root, width=100, show="*")
```

Al ejecutar e introducir texto será de tipo contraseña.

Vamos a presionar el botón.



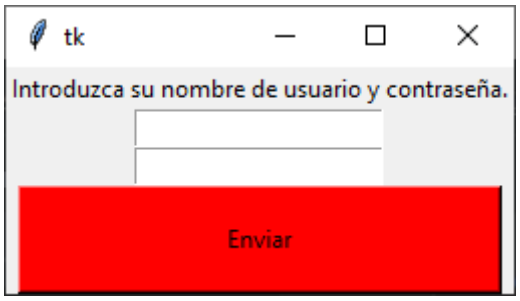
Los asteriscos que aparecen al principio es correspondiente a la siguiente línea.

```
entrada.insert(0, "Escriba aquí...")
```

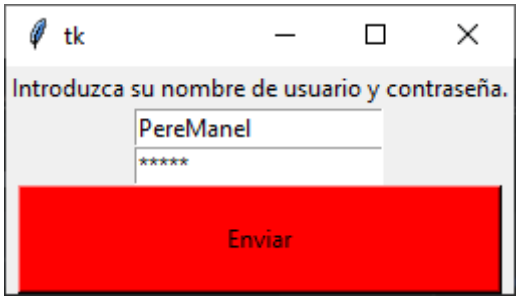
Si la eliminamos y ejecutamos de nuevo ya no aparecerán los asteriscos.

Ejercicio práctico:

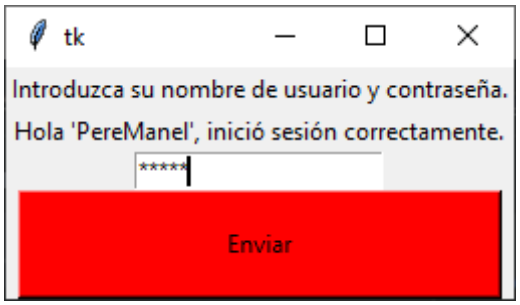
Al ejecutar queremos mostrar el siguiente formulario:



Como nombre de usuario PereManel y como contraseña 12345.



Vamos a presionar el botón.



Solución:

```
from tkinter import *
root = Tk()

texto1 = Label(root, text="Introduzca su nombre de usuario y
contraseña.").grid(row=0, column=0)

entrada1 = Entry(root, width=25)
entrada1.grid(row=1, column=0)

entrada2 = Entry(root, width=25, show="*")
entrada2.grid(row=2, column=0)

def click_boton():
    texto = Label(root, text=f"Hola '{entrada1.get()}', inició sesión
correctamente.").grid(row=1, column=0)

boton1 = Button(root, text="Enviar", bg="red", padx=100, pady=15,
command=click_boton).grid(row=3, column=0)

root.mainloop()
```

Capítulo 7: Con el widget Radiobutton() y VARIABLES de CONTROL

La variables de control son objetos especiales asociados a los widgets de Tkinter para almacenar sus valores y de esta forma poder trabajar con los distintos tipos de formularios que tiene.

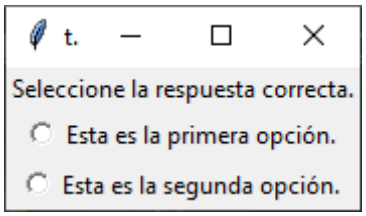
Tenemos variables de tipo numérico para los enteros, llamadas IntVar() para los enteros, DoubleVar() para los Float, para los String StringVar() y también para los boléanos llamda BooleanVar().

Las variables de control las vamos a utilizar para conectar varios widgets del mismo tipo, es decir si tenemos varias opciones de Radiobutton() podremos relacionarla a posibles opciones, creando así un grupo de widget.

El widget Radiobutton.

```
from tkinter import *
root = Tk()

# Creamos nuestra variable de control
x = IntVar()

titulo = Label(root, text="Seleccione la respuesta
correcta.").grid(row=0)

# la variable=x nos indica que es un grupo.
opcion_1 = Radiobutton(root, text="Esta es la primera opción.", value=1,
variable=x).grid(row=1)
opcion_2 = Radiobutton(root, text="Esta es la segunda opción.", value=2,
variable=x).grid(row=2)

root.mainloop()
```

Si lo ejecutamos:



Podremos seleccionar una de las dos opciones.

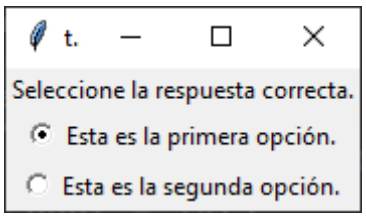
Si seleccionamos la primera opción la variable x tendrá el valor 1 y si seleccionamos la segunda opción la variable x tendrá el valor de 2.

Si te fijas no aparece ninguna opción por defecto activada.

```
from tkinter import *
root = Tk()

# Creamos nuestra variable de control
x = IntVar()
```

```
# Seleccionamos la primera opción por defecto.
x.set(value=1)

titulo = Label(root, text="Seleccione la respuesta
correcta.").grid(row=0)

# la variable=x nos indica que es un grupo.
opcion_1 = Radiobutton(root, text="Esta es la primera opción.", value=1,
variable=x).grid(row=1)
opcion_2 = Radiobutton(root, text="Esta es la segunda opción.", value=2,
variable=x).grid(row=2)

root.mainloop()
```

Vamos a ejecutar de nuevo:



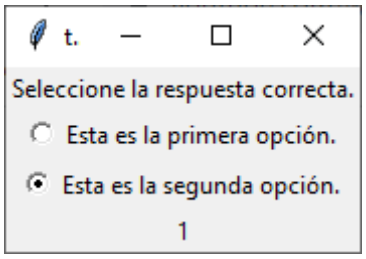
Pudiendo cambiar la opción.

El problema es que podemos cambiar de opción pero el valor que nos enviará el formulario siempre será el de 1.

Vamos a demostrarlo con un get.

```
from tkinter import *
root = Tk()

# Creamos nuestra variable de control
x = IntVar()
# Seleccionamos la primera opción por defecto.
x.set(value=1)
opcion_set = Label(root, text=x.get()).grid(row=3)

titulo = Label(root, text="Seleccione la respuesta
correcta.").grid(row=0)

# la variable=x nos indica que es un grupo.
opcion_1 = Radiobutton(root, text="Esta es la primera opción.", value=1,
variable=x).grid(row=1)
opcion_2 = Radiobutton(root, text="Esta es la segunda opción.", value=2,
variable=x).grid(row=2)

root.mainloop()
```

Vamos a ejecutar:



Si seleccionamos la segunda opción nos sigue mostrando el valor 1 y tendría que demostrar el valor 2.

Para que se pueda actualizar vamos a crear una función.

```
from tkinter import *
root = Tk()

# Creamos nuestra variable de control
x = IntVar()
# Seleccionamos la primera opción por defecto.
x.set(value=1)

def actualiza(value):
    opcion_set = Label(root, text=x.get()).grid(row=3)

titulo = Label(root, text="Seleccione la respuesta
correcta.").grid(row=0)

# la variable=x nos indica que es un grupo.
opcion_1 = Radiobutton(root, text="Esta es la primera opción.", value=1,
variable=x, command=lambda: actualiza(x.get())).grid(row=1)
opcion_2 = Radiobutton(root, text="Esta es la segunda opción.", value=2,
variable=x, command=lambda: actualiza(x.get())).grid(row=2)

root.mainloop()
```

Creamos una función para que se puedan actualizar los datos.

La función se encarga de mostrar en un “Label()” el “value” obtenido en el “get” de las funciones lambda pasadas como evento en el “command” de los “Radiobutton()”.

Desde las opción_1 y opción_2 llamamos a la función actualiza(x.get() con el método lambda.

Si eliminamos la opción lambda esta función solo se ejecutar al principio y cuando cambiemos de opción no se ejecutará.

Capítulo 8: Bucle AUTOGENERADOR de Radiobuttons y botón de envío

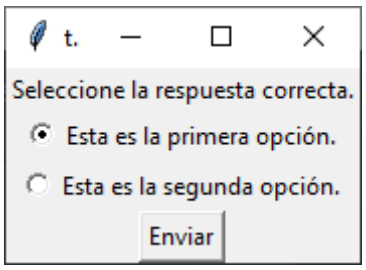
Siguiendo con el capitulo anterior queremos que cambie de valor cuando presionemos al botón enviar.

```
from tkinter import *
root = Tk()

# Creamos nuestra variable de control
x = IntVar()
# Seleccionamos la primera opción por defecto.
x.set(value=1)

def actualiza(value):
    opcion_set = Label(root, text=x.get()).grid(row=3)

titulo = Label(root, text="Seleccione la respuesta
correcta.").grid(row=0)

# la variable=x nos indica que es un grupo.
opcion_1 = Radiobutton(root, text="Esta es la primera opción.", value=1,
variable=x).grid(row=1)
opcion_2 = Radiobutton(root, text="Esta es la segunda opción.", value=2,
variable=x).grid(row=2)

boton_envia = Button(root, text="Enviar", command=lambda :
actualiza(x.get())).grid(row=4)

root.mainloop()
```

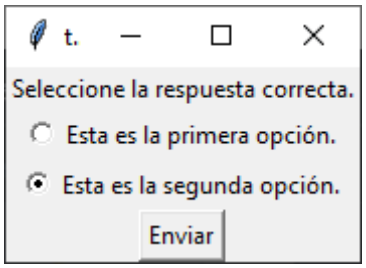
La opción command con el método lambda las borramos de las variables opcion_1 y opcion_2.

Creamos un objeto botón:envia de tipo Button y agregamos la sentencia command, ya que al presionar el botón este ejecutará la función actualiza(value).

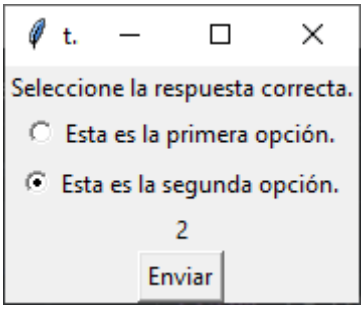
Vamos a ejecutar:

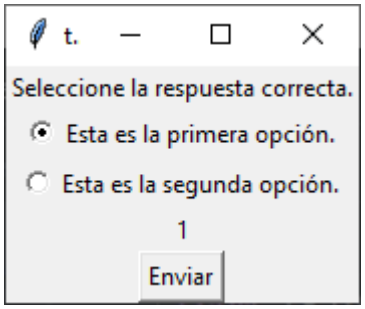


Ahora seleccionaremos la segunda opción.



Hacemos clic en el botón Enviar.





Puedes cambiar a la opción 1 y presionar de nuevo el botón enviar.

Ahora pasa el valor de la variable x cuando presionamos el botón Enviar.

Ahora vamos a utilizar un ciclo for para agregar varios Radiobutton().

```
from tkinter import *
root = Tk()

def actualiza(value):
    opcion_set = Label(root, text=value).pack()

titulo = Label(root, text="Seleccione un opción.").pack()

opciones = [["Color rojo", "rojo"],
            ["Color azul", "azul"],
            ["Color verde", "verde"],
            ["Color amarillo", "amarillo"]]

colores = StringVar()
colores.set("rojo")

for opcion, valor in opciones:
    Radiobutton(root, text=opcion, value=valor, variable=colores).pack()

boton_envia = Button(root, text="Enviar",
command=lambda:actualiza(colores.get())).pack()
root.mainloop()
```

Creamos una lista, que a su vez contendrá 4 listas más con los respectivos colores que serás las opciones que tenemos que elegir.

Definimos colores como una variable de control para almacenar Strings. Le asignamos por defecto el color “rojo”.

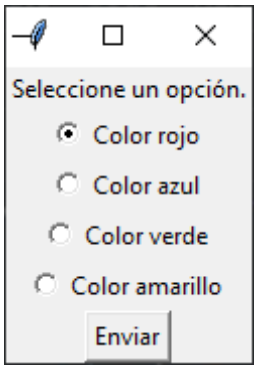
En un ciclo for que iterará tantas veces como los elementos que tiene la lista opciones (4), ya que contine a su ver 4 listas más de dos elementos.

Utilizamos .pack() para que los vaya colocando una debajo del otro.

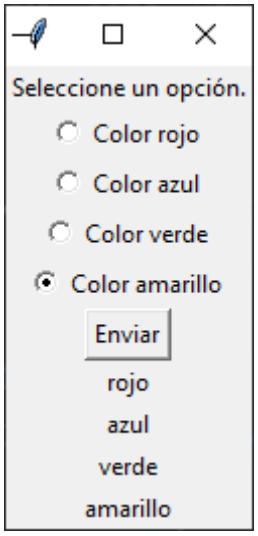
Definimos el botón_envia como objeto Button con el texto de “Enviar”, cuando ahamos clic ejecutará la función Actualiza(colores.get), como parámetro envia el valor de colores.get, este puede ser rojo, azul, verde o amarillo.

Con .pack() La etiqueta obtenida con su respectivo color lo coloca en la parte inferior del formulario.

Ahora vamos a ejejcutar:



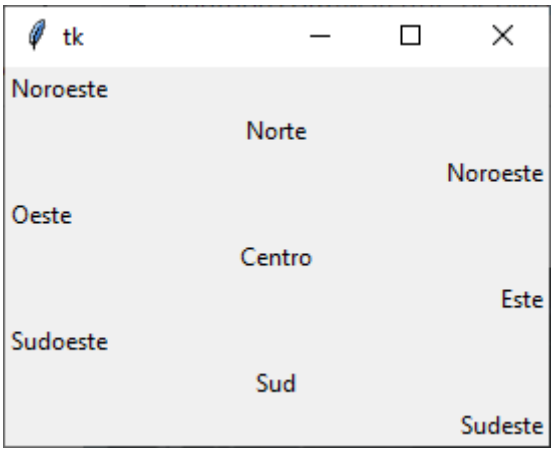
Ahora selecciona cada color seguido del botón enviar.



Capítulo 9: Los anclajes de Tkinter

```
from tkinter import *
root = Tk()

titulo1 = Label(root, text="Noroeste").pack(anchor=NW)
titulo2 = Label(root, text="Norte").pack(anchor=N)
titulo3 = Label(root, text="Noroeste").pack(anchor=NE)
titulo4 = Label(root, text="Oeste").pack(anchor=W)
titulo5 = Label(root, text="Centro").pack(anchor=CENTER)
titulo6 = Label(root, text="Este").pack(anchor=E)
titulo7 = Label(root, text="Sudoeste").pack(anchor=SW)
titulo8 = Label(root, text="Sud").pack(anchor=S)
titulo9 = Label(root, text="Sudeste").pack(anchor=SE)

root.mainloop()
```

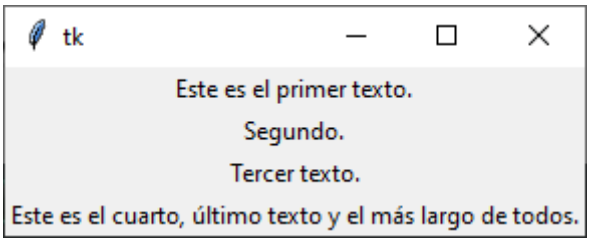
Si ejecutamos este será el resultado:



Vamos a ver cómo utilizarlo.

```
from tkinter import *
root = Tk()

titulo1 = Label(root, text="Este es el primer texto.").pack()
titulo2 = Label(root, text="Segundo.").pack()
titulo3 = Label(root, text="Tercer texto.").pack()
titulo4 = Label(root, text="Este es el cuarto, último texto y el más
largo de todos.").pack()

root.mainloop()
```

Si lo ejecutamos:

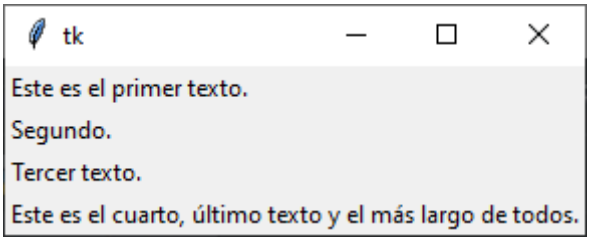


Vamos a modificarlo para que todos los textos estén alineados hacia la izquierda.

```
from tkinter import *
root = Tk()

titulo1 = Label(root, text="Este es el primer texto.").pack(anchor=NW)
titulo2 = Label(root, text="Segundo.").pack(anchor=NW)
titulo3 = Label(root, text="Tercer texto.").pack(anchor=NW)
titulo4 = Label(root, text="Este es el cuarto, último texto y el más
largo de todos.").pack(anchor=NW)

root.mainloop()
```

después de .pack(anchor=NW)

Este será el resultado:



Capítulo 10: Los cuadros de diálogo (MESSAGERBOX)

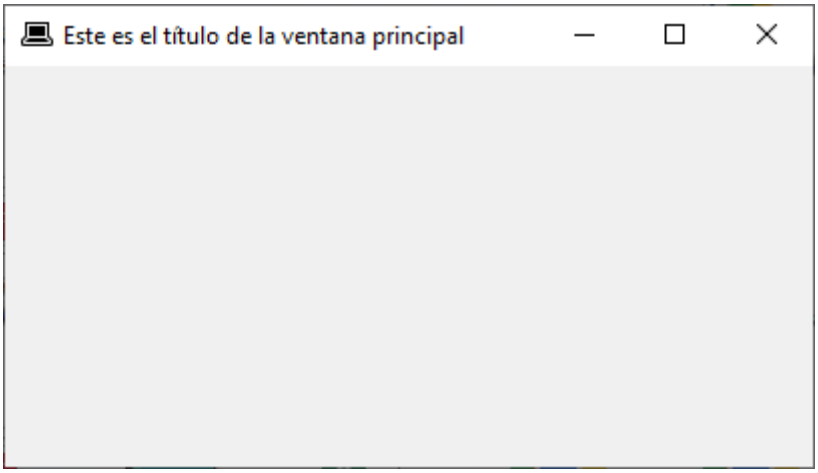
En este capítulo vamos a aprender como hacer cuadros de diálogo.

Para realizar este proyecto crear una carpeta donde tienen es documento actual de Python y crear una carpeta llamada img, copia en ella un archivo de tipo icono en nuestro caso se llama pc.ico.

```
from tkinter import *
from tkinter.messagebox import *

root = Tk()

root.title("Este es el título de la ventana principal")
root.iconbitmap('img/pc.ico')

root.mainloop()
```

Vamos a ver como se crear un título en la ventana principal con el método title y agregar un icono con el método iconbitmap.

Cuando ejecutemos este será el resultado:



Ahora vamos a realizar los pasos necesarios para mostrar un cuadro de diálogo.

Vamos a crear un función que cuando realicemos un evento como pulsar un botón se muestre un cuadro de diálogo.

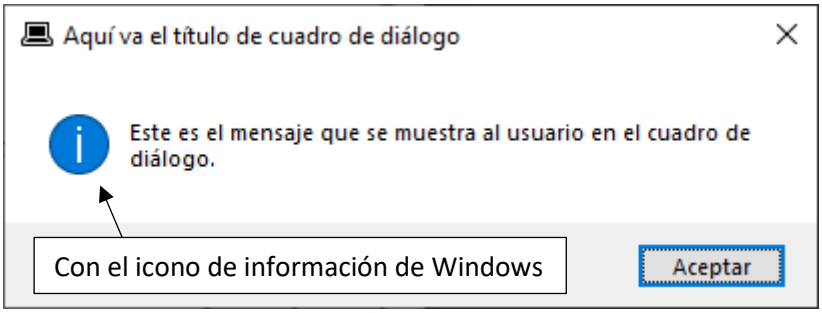
```
from tkinter import *
from tkinter.messagebox import *
root = Tk()
root.title("Este es el título de la ventana principal")
root.iconbitmap('img/pc.ico')

def muestra_ventana():
    showinfo("Aquí va el título de cuadro de diálogo", "Este es el
mensaje que se muestra al usuario en el cuadro de diálogo.")
```

```
boton1 = Button(root, text="Enviar", command=muestra_ventana,
width=75).pack()

root.mainloop()
```

Creamos una función llamada muestra_ventana() llamamos a showinfo en un tipo de cuadro de diálogo(“Titulo del cuadro de diálogo”, “Mensaje”). Creamos un boton1 objeto de Button, con su texto “Enviar” y cada vez que hacemos clic ejecutamos la función muestra_ventana, con un ancho de 75 px. Y un pack para posicionarlo.

Vamos a ejecutar:

Vamos a presionar el botón Enviar:



Para cerrar esta ventana le damos al botón Aceptar.

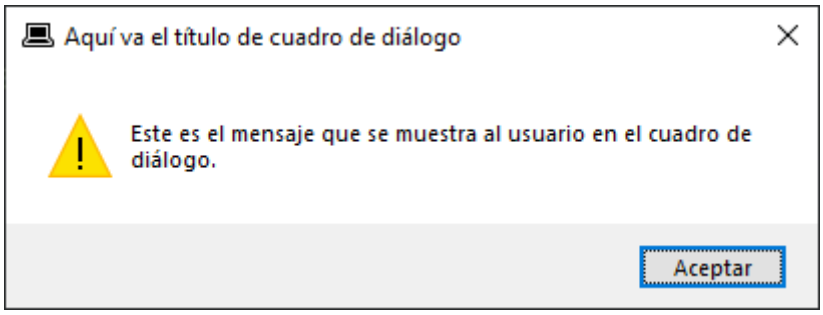
```
showinfo(title="Aquí va el título de cuadro de diálogo", message="Este es
el mensaje que se muestra al usuario en el cuadro de diálogo.")
```

Podemos definir lo que es el título y el mensaje como se muestra en el ejemplo, como están identificados podemos invertir el orden, poner primero el mensaje y a continuación el título.
Hay otros tipos de cuadros de diálogo, showwarning().

```
showwarning(title="Aquí va el título de cuadro de diálogo", message="Este
es el mensaje que se muestra al usuario en el cuadro de diálogo.")
```

Observaremos que lo que cambia es el icono.
Vamos a ejecutar.

Presionamos el botón Enviar.

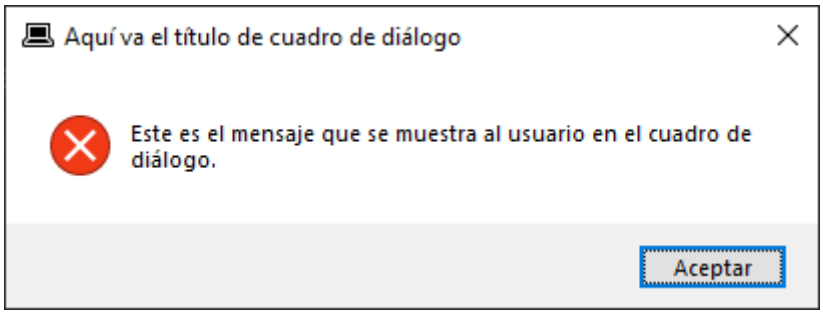


Para los errores podemos utilizar un showerror().

```
showerror(title="Aquí va el título de cuadro de diálogo", message="Este es el mensaje que se muestra al usuario en el cuadro de diálogo.")
```

Vamos a ejecutar:

Presionamos el botón Enviar.

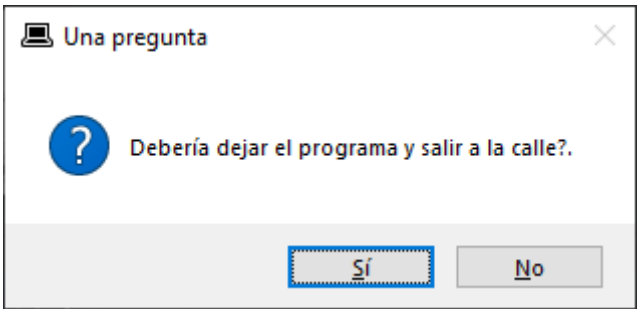


Ahora vamos a ver los cuadros de diálogo que te formula una pregunta.

askquestion, askyesno, askretrycancel, askkyesnocancel y askokcancel.

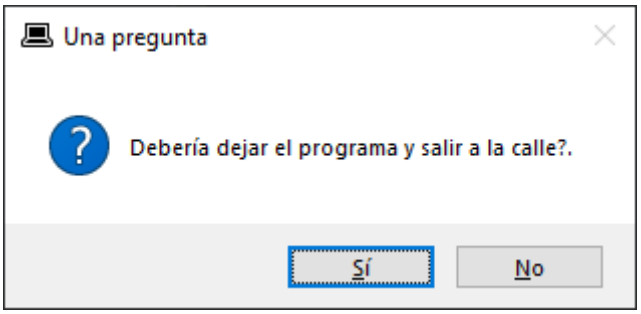
Vamos a empezar con askquestion().

```
askquestion(title="Una pregunta", message="Debería dejar el programa y salir a la calle?.")
```



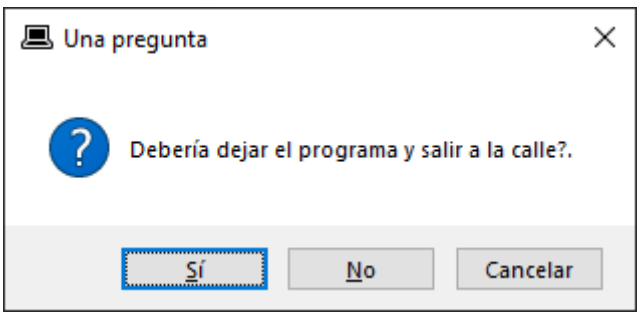
Ahora vamos con askyesno().

```
askyesno(title="Una pregunta", message="Debería dejar el programa y salir a la calle?.")
```



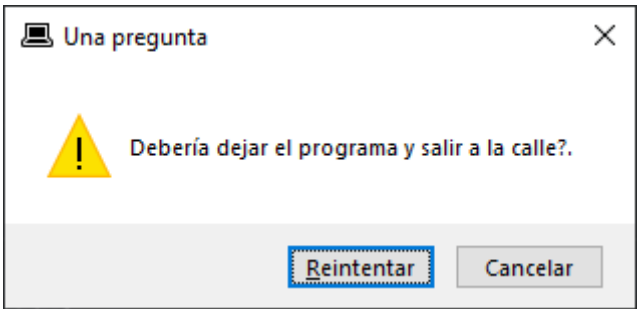
Ahora vamos con askesnocancel().

```
askyesnocancel(title="Una pregunta", message="Debería dejar el programa y salir a la calle?.")
```



Ahora vamos askretrycancel()

```
askretrycancel(title="Una pregunta", message="Debería dejar el programa y salir a la calle?.")
```


Una pregunta
Debería dejar el programa y salir a la calle?.
Reintentar
Cancelar

Capítulo 11: Añadir código a las opciones de los MESSAGEBOX

```
from tkinter import *
from tkinter.messagebox import *

root = Tk()

root.title("Este es el título de la ventana principal")
root.iconbitmap('img/pc.ico')

def muestra_ventana():
    respuesta = askquestion(title="Pregunta seria", message="Debería
dejar el programa y salir a la calle?.")
    if respuesta == "no":
        showinfo(title="¡A segujir programando!", message="Estupendo,
eligió la respuesta correcta.")
    else:
        askretrycancel(title="Botón equivoado",message="Haga click en
'Reintenar' para seguir programando.")

boton1 = Button(root, text="Enviar", command=muestra_ventana,
width=75).pack()
root.mainloop()
```

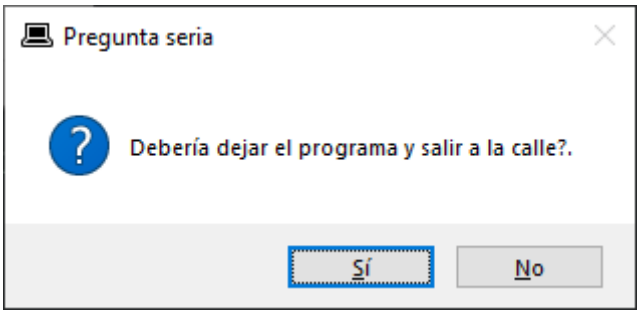
Vamos a comentar la función muestra_ventana:

Hay unos cuadros de diálogo que nos realiza una pregunta y tu tienes que contestar con unas opciones que están en botones. Según la opción que elijas retorna un valor, en este ejemplo lo almacenamos en la variable respuesta, que podrá obtener los valores yes o no.

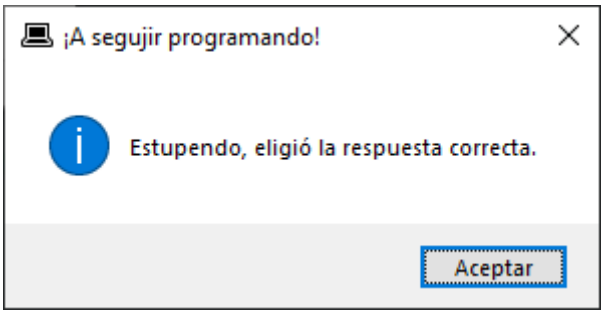
A partir de esta respuesta con un condicional comparamos si la respuesta es “no” no esnviará un showinfo(), de lo contrario nos enviará askretrycancel()

Este será el resultado cuando ejecutemos:

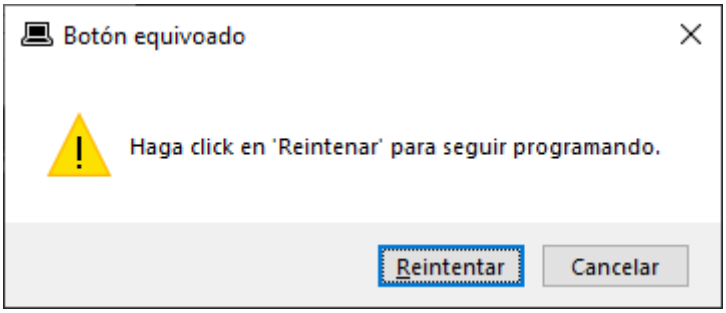
Presionamos el botón Enviar.



Vamos a contestar que no.



En el caso contrario si hubiéramos contestado que si este sería el resultado:



En este caso deberías agregar otra opción con su respectiva variable para controlar la respuesta del usuario.

Otra forma de realizarlo sería:

```
def muestra_ventana():
    respuesta = askquestion(title="Pregunta seria", message="Debería
dejar el programa y salir a la calle?.")
    if respuesta == "no":
        showinfo(title="¡A segujir programando!", message="Estupendo,
eligió la respuesta correcta.")
    if respuesta == "yes":
        askretrycancel(title="Botón equivoado",message="Haga click en
'Reintenar' para seguir programando.")
```

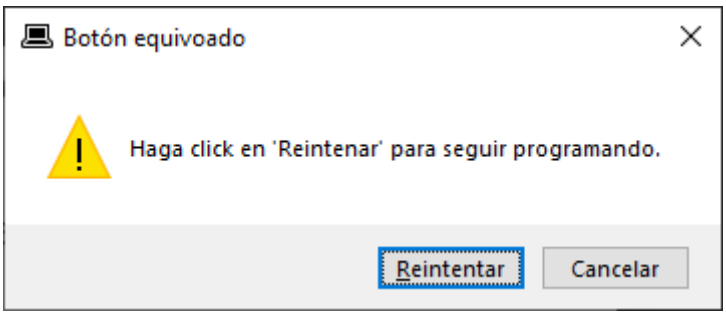
Vamos a comprobar si funciona y además confirmamos que al selección Sí retorna un yes.

Vamos a modificar la función def muestra_ventana():

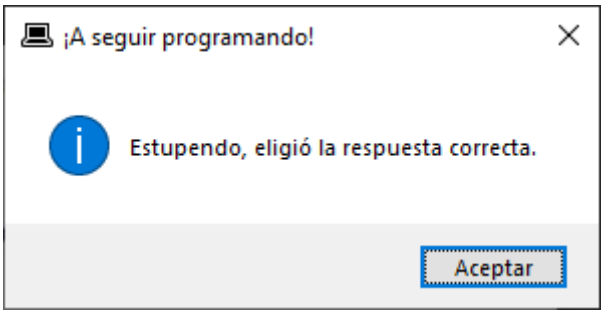
```
def muestra_ventana():
    respuesta = askquestion(title="Pregunta seria", message="Debería
dejar el programa y salir a la calle?.")
    if respuesta == "no":
        showinfo(title="¡A segujir programando!", message="Estupendo,
eligió la respuesta correcta.")
    if respuesta == "yes":
        respuesta_retry = askretrycancel(title="Botón
equivoado",message="Haga click en 'Reintenar' para seguir programando.")
        if respuesta_retry: # Retorna un valor boleano.
```

```
            showinfo(title="¡A seguir programando!", message="Estupendo,
eligió la respuesta correcta.")
```

Si seleccionamos el botón Reintentar: nos retornará un True.



Nos mostrará el siguiente cuadro de diálogo:



Si en lugar de Reintentar seleccionamos Cancelar nos retornará un False.

```
if not respuesta_retry: # Compara si es False.
            showinfo(title="¡A seguir programando!", message="Estupendo,
eligió la respuesta correcta.")
```

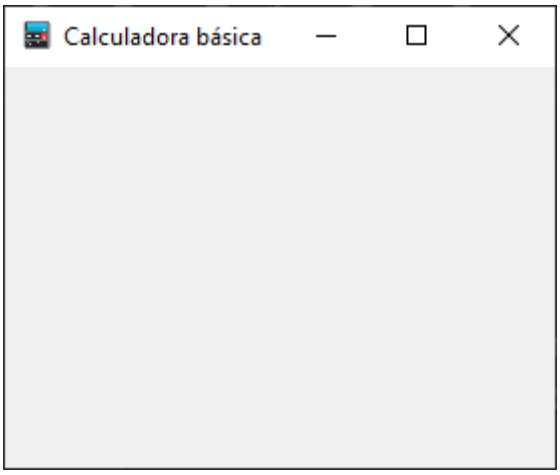
Como último ejemplo vamos a utilizar un else.

```
def muestra_ventana():
    respuesta = askquestion(title="Pregunta seria", message="Debería
dejar el programa y salir a la calle?.")
    if respuesta == "no":
        showinfo(title="¡A segujir programando!", message="Estupendo,
eligió la respuesta correcta.")
    if respuesta == "yes":
        respuesta_retry = askretrycancel(title="Botón
equivoado",message="Haga click en 'Reintenar' para seguir programando.")
        if respuesta_retry: # Retorna un valor boleano.
            showinfo(title="¡A seguir programando!", message="Estupendo,
eligió la respuesta correcta.")
        else:
             showinfo(title="¡Adios...", message="Qué tengas un buen día
T.T.")
```

Capítulo 12: Cómo crear una CALCULADORA – parte gráfica

```
from tkinter import *

root = Tk()
root.title("Calculadora básica")
root.iconbitmap("img/calculadora.ico")

mainloop()
```

empezamos por la estructura básica, este será el resultado:



En este capítulos vamos a trabajar el entorno gráfico y en el siguiente capítulo la lógica.

Este será el código:

```
from tkinter import *

root = Tk()
root.title("Calculadora básica")
root.iconbitmap("img/calculadora.ico")
# Impide que el usuario modifique el tamaño de la ventana.
# Los argumentos se refiere a la x y el segundo a la y.
# Si el primer valor lo cambiamos a un 1 nos dejará modificar en
horizontal.
# Si el segundo valor lo cambiamos a un 1 nos dejará modificar en
vertical.
# Por defecto los valores son 1 y 1.
root.resizable(0,0)
# Para dar dimensiones a la ventana.
root.geometry("296x265")
# Caja de texto
pantalla =Entry(root, width=22, bg="black", fg="white", borderwidth=0,
font=('arial', 18, 'bold'))
# Posicionamos la caja de texto en la fila 0 dejando un marge de 2 px. y
que ocupe 4 col.
pantalla.grid(row=0, padx=2, pady=2, columnspan=4)
```

```
# Botones con los números
boton_1 = Button(root, text="1", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=1, column=0, padx=1, pady=1)

boton_2 = Button(root, text="2", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=1, column=1, padx=1, pady=1)

boton_3 = Button(root, text="3", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=1, column=2, padx=1, pady=1)

boton_4 = Button(root, text="4", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=2, column=0, padx=1, pady=1)

boton_5 = Button(root, text="5", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=2, column=1, padx=1, pady=1)

boton_6 = Button(root, text="6", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=2, column=2, padx=1, pady=1)

boton_7 = Button(root, text="7", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=3, column=0, padx=1, pady=1)

boton_8 = Button(root, text="8", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=3, column=1, padx=1, pady=1)

boton_9 = Button(root, text="9", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=3, column=2, padx=1, pady=1)

boton_0 = Button(root, text="0", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2").grid(row=4, column=1, padx=1, pady=1)

boton_igual = Button(root, text="=", width=9, height=3, bg="red",
fg="white", borderwidth=0, cursor="hand2").grid(row=4, column=0, padx=1,
pady=1)

boton_punto = Button(root, text=".", width=9, height=3, bg="spring
green", fg="black", borderwidth=0, cursor="hand2").grid(row=4, column=2,
padx=1, pady=1)

boton_mas = Button(root, text="+", width=9, height=3, bg="deep sky blue",
fg="black", borderwidth=0, cursor="hand2").grid(row=1, column=3, padx=1,
pady=1)

boton_menos = Button(root, text="-", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2").grid(row=2, column=3,
padx=1, pady=1)
```

```
boton_multiplicacion = Button(root, text="*", width=9, height=3, bg="deep
sky blue", fg="black", borderwidth=0, cursor="hand2").grid(row=3,
column=3, padx=1, pady=1)

boton_division = Button(root, text="/", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2").grid(row=4, column=3,
padx=1, pady=1)

mainloop()
```

Este será el resultado final:

Capítulo 13: Cómo crear una CALCULADORA – parte lógica

Vamos a crear la siguiente función:

```
def envia_boton(valor):
    anterior = pantalla.get()
    pantalla.delete(0, END)  # Borra la pantalla
    pantalla.insert(0, str(anterior) + str(valor))
```

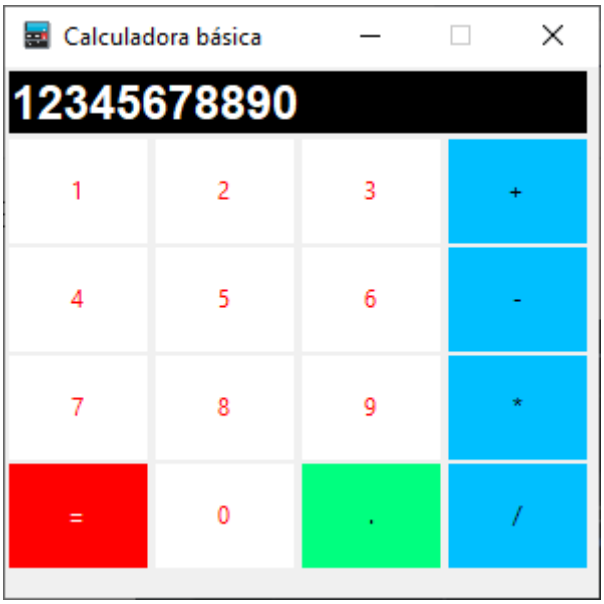
Ahora vamos a ver el código del botón_1:

```
boton_1 = Button(root, text="1",
      width=9,
      height=3,
      bg="white",
      fg="red",
      borderwidth=0,
      cursor="hand2",
      command=lambda: envia_boton(1)).grid(row=1, column=0, padx=1,
pady=1)
```

Llamamos a la función pasándole el parámetro 1.

Vamos a ejecutar la calculadora y pulsar varias veces el número 1.

Hay q que poner el command=lambda: envia_boton(num) en el resto de números, para num será el numero que tiene el botón.



Vamos a crear las funciones, una para la suma, otra para la resta, otra para la multiplicación y otra para la división.

```
def envia_boton(valor):
    anterior = pantalla.get()
    pantalla.delete(0, END)  # Borra la pantalla
    pantalla.insert(0, str(anterior) + str(valor))

def igual():
    global num2
    num2 = pantalla.get()
    pantalla.delete(0, END)
    if operacion == "+":
        pantalla.insert(0, float(num1) + float(num2))
    if operacion == "-":
        pantalla.insert(0, float(num1) - float(num2))
    if operacion == "*":
        pantalla.insert(0, float(num1) * float(num2))
    if operacion == "/":
        pantalla.insert(0, float(num1) / float(num2))

def suma():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "+"

def resta():
    global num1
    global operacion
    num1 = pantalla.get()
```

```
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "-"

def multiplicacion():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "*"

def division():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "/"
```

Vamos a llamar a las funciones:

```
boton_1 = Button(root, text="1", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(1)).grid(row=1, column=0, padx=1, pady=1)

boton_2 = Button(root, text="2", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(2)).grid(row=1, column=1, padx=1, pady=1)

boton_3 = Button(root, text="3", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(3)).grid(row=1, column=2, padx=1, pady=1)

boton_4 = Button(root, text="4", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(4)).grid(row=2, column=0, padx=1, pady=1)

boton_5 = Button(root, text="5", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(5)).grid(row=2, column=1, padx=1, pady=1)

boton_6 = Button(root, text="6", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(6)).grid(row=2, column=2, padx=1, pady=1)
```

```
boton_7 = Button(root, text="7", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(7)).grid(row=3, column=0, padx=1, pady=1)

boton_8 = Button(root, text="8", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(8)).grid(row=3, column=1, padx=1, pady=1)

boton_9 = Button(root, text="9", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(9)).grid(row=3, column=2, padx=1, pady=1)

boton_0 = Button(root, text="0", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(0)).grid(row=4, column=1, padx=1, pady=1)
```

Para los botones que contienen números llamaremos a la función **command=lambda: enviar_boton(número_del_boton).**

Ahora para el botón que contiene el punto (.):

```
boton_punto = Button(root, text=".", width=9, height=3, bg="spring
green", fg="black", borderwidth=0, cursor="hand2", command=lambda:
envia_boton(".")).grid(row=4, column=2, padx=1, pady=1)
```

Ahora la función para el botón igual.

```
boton_igual = Button(root, text="=", width=9, height=3, bg="red",
fg="white", borderwidth=0, cursor="hand2", command=lambda:
igual()).grid(row=4, column=0, padx=1, pady=1)
```

Llamaremos a la función igual()

Ahora para la función de los operadores de suma, resta, multiplicación y división:

```
boton_mas = Button(root, text="+", width=9, height=3, bg="deep sky blue",
fg="black", borderwidth=0, cursor="hand2", command=lambda:
suma()).grid(row=1, column=3, padx=1, pady=1)
```

La función command=lambda: suma()

```
boton_menos = Button(root, text="-", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
resta()).grid(row=2, column=3, padx=1, pady=1)
```

La función command=lambda: resta()

```
boton_multiplicacion = Button(root, text="*", width=9, height=3, bg="deep
sky blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
multiplicacion()).grid(row=3, column=3, padx=1, pady=1)
```

La función command=lambda: muiltiplicacion()

```
boton_division = Button(root, text="/", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
division()).grid(row=4, column=3, padx=1, pady=1)
```

La función command=lambda: division()

Nos falta un botón para borrar la pantalla.

```
boton_despejar = Button(root,
width=40,
height=3,
text="Borrar",
bg="deep sky blue",
fg="black",
borderwidth=0,cursor="hand2",
command=lambda: despejar()).grid(row=5,column=0, columnspan=4, padx=1,
pady=1)
```

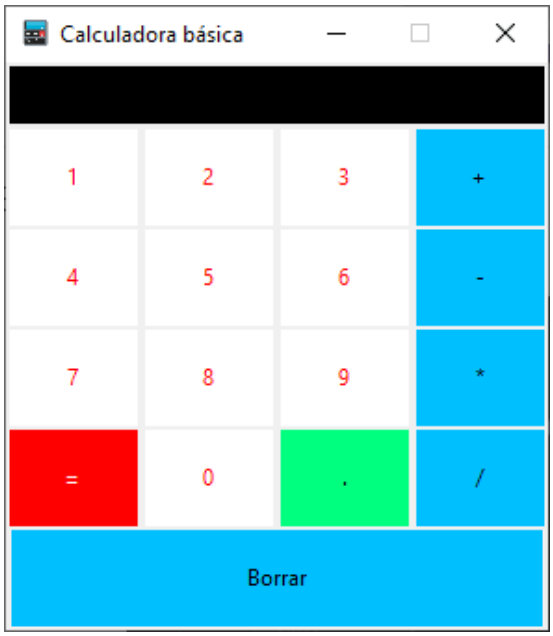
Asi como su función:

```
def despejar():
    pantalla.delete(0, END)
```

También podemos ajustar las dimensiones de la ventana:

```
# Para dar dimensiones a la ventana.
root.geometry("292x306")
```

Este será el resultado final:



Relación de todo el código:

```
from tkinter import *

root = Tk()
root.title("Calculadora básica")
root.iconbitmap("img/calculadora.ico")
# Impide que el usuario modifique el tamaño de la ventana.
# Los argumentos se refiere a la x y el segundo a la y.
# Si el primer valor lo cambiamos a un 1 nos dejará modificar en
horizontal.
# Si el segundo valor lo cambiamos a un 1 nos dejará modificar en
vertical.
# Por defecto los valores son 1 y 1.
root.resizable(0,0)
# Para dar dimensiones a la ventana.
root.geometry("292x306")

def envia_boton(valor):
    anterior = pantalla.get()
    pantalla.delete(0, END)  # Borra la pantalla
    pantalla.insert(0, str(anterior) + str(valor))

def igual():
    global num2
    num2 = pantalla.get()
    pantalla.delete(0, END)
    if operacion == "+":
        pantalla.insert(0, float(num1) + float(num2))
    if operacion == "-":
        pantalla.insert(0, float(num1) - float(num2))
    if operacion == "*":
```

```
            pantalla.insert(0, float(num1) * float(num2))
        if operacion == "/":
            pantalla.insert(0, float(num1) / float(num2))

def suma():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "+"

def resta():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "-"

def multiplicacion():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "*"

def division():
    global num1
    global operacion
    num1 = pantalla.get()
    num1 = float(num1)
    pantalla.delete(0, END)
    operacion = "/"

def despejar():
    pantalla.delete(0, END)


# Caja de texto
pantalla =Entry(root, width=22, bg="black", fg="white", borderwidth=0,
font=('arial', 18, 'bold'))

# Posicionamos la caja de texto en la fila 0 dejando un marge de 2 px. y
que ocupe 4 col.
pantalla.grid(row=0, padx=2, pady=2, columnspan=4)

# Botones con los números
```

```
boton_1 = Button(root, text="1", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(1)).grid(row=1, column=0, padx=1, pady=1)

boton_2 = Button(root, text="2", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(2)).grid(row=1, column=1, padx=1, pady=1)

boton_3 = Button(root, text="3", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(3)).grid(row=1, column=2, padx=1, pady=1)

boton_4 = Button(root, text="4", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(4)).grid(row=2, column=0, padx=1, pady=1)
boton_5 = Button(root, text="5", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(5)).grid(row=2, column=1, padx=1, pady=1)

boton_6 = Button(root, text="6", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(6)).grid(row=2, column=2, padx=1, pady=1)

boton_7 = Button(root, text="7", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(7)).grid(row=3, column=0, padx=1, pady=1)

boton_8 = Button(root, text="8", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(8)).grid(row=3, column=1, padx=1, pady=1)

boton_9 = Button(root, text="9", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(9)).grid(row=3, column=2, padx=1, pady=1)

boton_0 = Button(root, text="0", width=9, height=3, bg="white", fg="red",
borderwidth=0, cursor="hand2", command=lambda:
envia_boton(0)).grid(row=4, column=1, padx=1, pady=1)

boton_igual = Button(root, text="=", width=9, height=3, bg="red",
fg="white", borderwidth=0, cursor="hand2", command=lambda:
igual()).grid(row=4, column=0, padx=1, pady=1)

boton_punto = Button(root, text=".", width=9, height=3, bg="spring
green", fg="black", borderwidth=0, cursor="hand2", command=lambda:
envia_boton(".")).grid(row=4, column=2, padx=1, pady=1)
```

```
boton_mas = Button(root, text="+", width=9, height=3, bg="deep sky blue",
fg="black", borderwidth=0, cursor="hand2", command=lambda:
suma()).grid(row=1, column=3, padx=1, pady=1)

boton_menos = Button(root, text="-", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
resta()).grid(row=2, column=3, padx=1, pady=1)

boton_multiplicacion = Button(root, text="*", width=9, height=3, bg="deep
sky blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
multiplicacion()).grid(row=3, column=3, padx=1, pady=1)

boton_division = Button(root, text="/", width=9, height=3, bg="deep sky
blue", fg="black", borderwidth=0, cursor="hand2", command=lambda:
division()).grid(row=4, column=3, padx=1, pady=1)

boton_despejar = Button(root, width=40, height=3, text="Borrar", bg="deep
sky blue", fg="black", borderwidth=0,cursor="hand2", command=lambda:
despejar()).grid(row=5,column=0, columnspan=4, padx=1, pady=1)

mainloop()
```

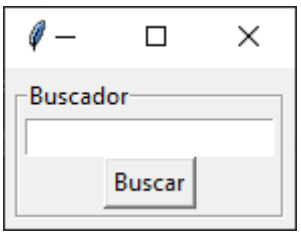
Queremos controlar un error si le damos al signo igual sin no haber realizado una operación que muestre en pantalla ¡ERROR!.

```
def igual():
    try:
        global num2
        num2 = pantalla.get()
        pantalla.delete(0, END)
        if operacion == "+":
            pantalla.insert(0, float(num1) + float(num2))
        if operacion == "-":
            pantalla.insert(0, float(num1) - float(num2))
        if operacion == "*":
            pantalla.insert(0, float(num1) * float(num2))
        if operacion == "/":
            pantalla.insert(0, float(num1) / float(num2))
    except NameError:
        pantalla.insert(0, "Error")
```

Capítulo 14: El widget LabelFrame

```
from tkinter import *

root = Tk()
root.title("Frames")

buscador = LabelFrame(root, text="Buscador", padx=3, pady=3)
buscador.grid(row=0, column=0, padx=5, pady=5)
barra = Entry(buscador, text="¿Buscas algo?").pack()
boton = Button(buscador, text="Buscar").pack()

mainloop()
```

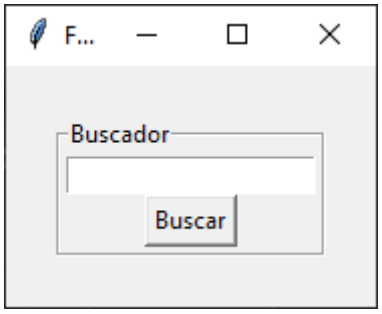
Este será el resultado:



Vamos a cambiar algunos parámetros:

```
from tkinter import *

root = Tk()
root.title("Frames")

buscador = LabelFrame(root, text="Buscador", padx=3, pady=3)
buscador.grid(row=0, column=0, padx=25, pady=25)
barra = Entry(buscador, text="¿Buscas algo?").pack()
boton = Button(buscador, text="Buscar").pack()

mainloop()
```

Este será el resultado:

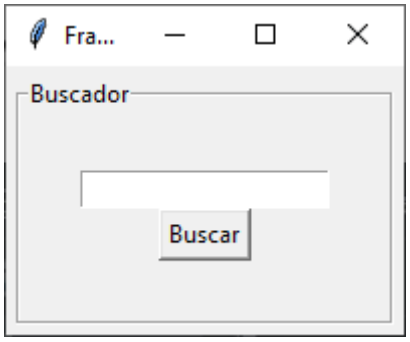


Los márgenes los crea en el exterior.

Vamos a cambiar otros parámetros:

```
from tkinter import *

root = Tk()
root.title("Frames")

buscador = LabelFrame(root, text="Buscador", padx=30, pady=30)
buscador.grid(row=0, column=0, padx=5, pady=5)
barra = Entry(buscador, text="¿Buscas algo?").pack()
boton = Button(buscador, text="Buscar").pack()

mainloop()
```

Este será el resultado:



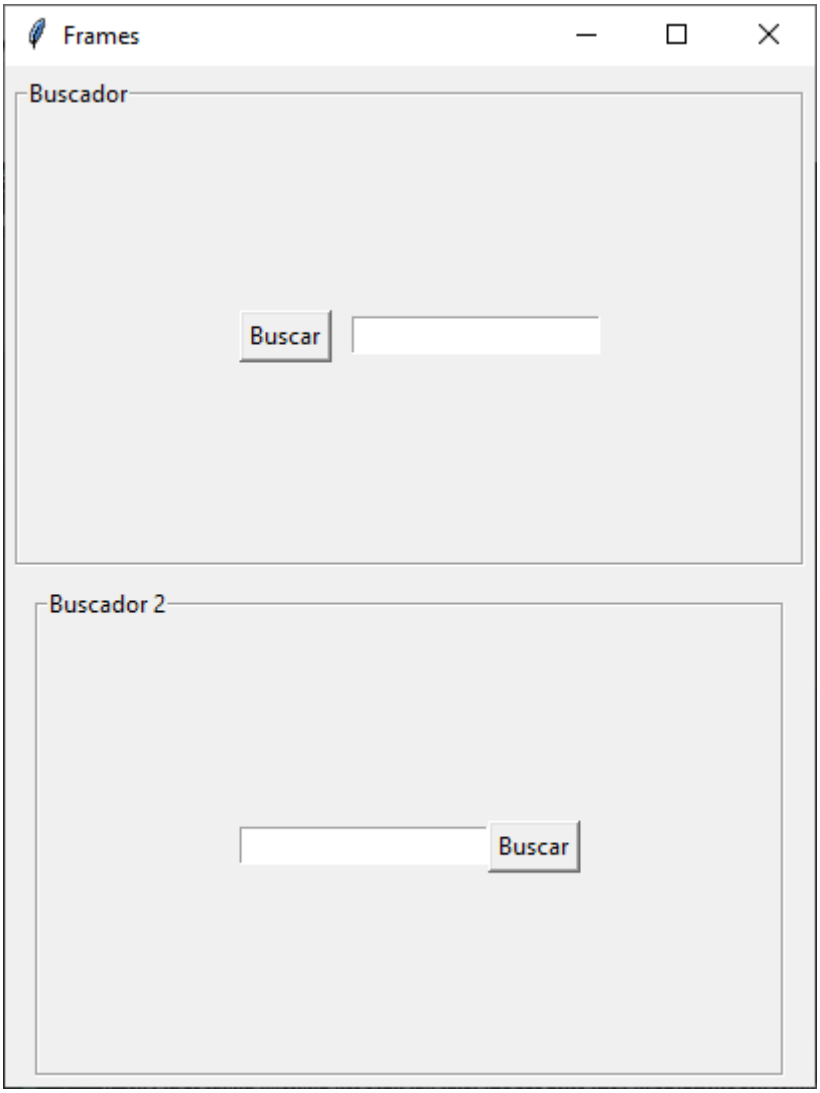
Los márgenes lo crea en el interior.

Vamos a ver otro ejemplo:

```
from tkinter import *

root = Tk()
root.title("Frames")

buscador = LabelFrame(root, text="Buscador", padx=100, pady=100)
buscador.grid(row=0, column=0, padx=5, pady=5)

barra = Entry(buscador, text="¿Buscas algo?").grid(row=0, column=1)
boton = Button(buscador, text="Buscar").grid(row=0, column=0, padx=10)

buscador2 = LabelFrame(root, text="Buscador 2", padx=100, pady=100)
buscador2.grid(row=1, column=0, padx=5, pady=5)

barra2 = Entry(buscador2, text="¿Buscas algo?").grid(row=0, column=0)
boton2 = Button(buscador2, text="Buscar").grid(row=0, column=1)
```

```
mainloop()
```

Este será el resultado:

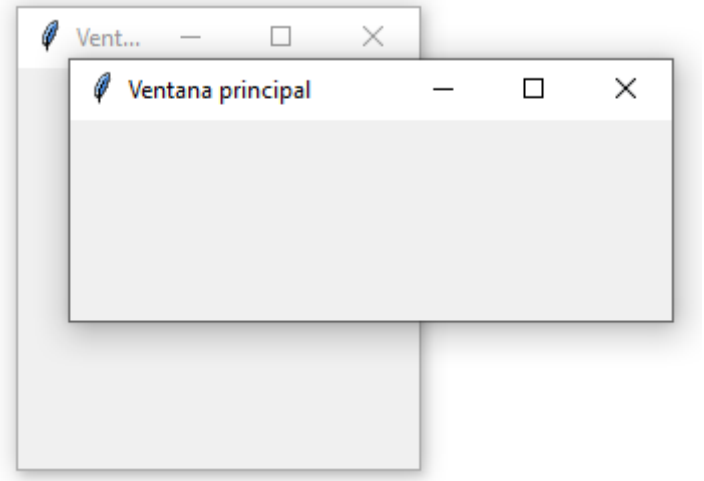


Capítulo 15: Nuevas ventanas con Toplevel()

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

ventana_nueva = Toplevel()

mainloop()
```

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

ventana_nueva = Toplevel()

mainloop()
```

Vamos a ejecutar:

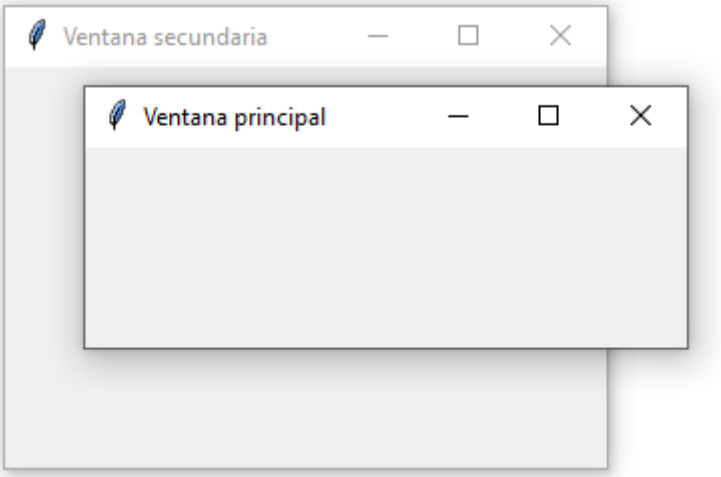


Se cerramos la segunda ventana la primera se mantendrá abierta pero si cerramos al ventana principal la segunda también se cerrar.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

ventana_nueva = Toplevel()
```

```
ventana_nueva.title("Ventana secundaria")
ventana_nueva.geometry("300x200")

mainloop()
```

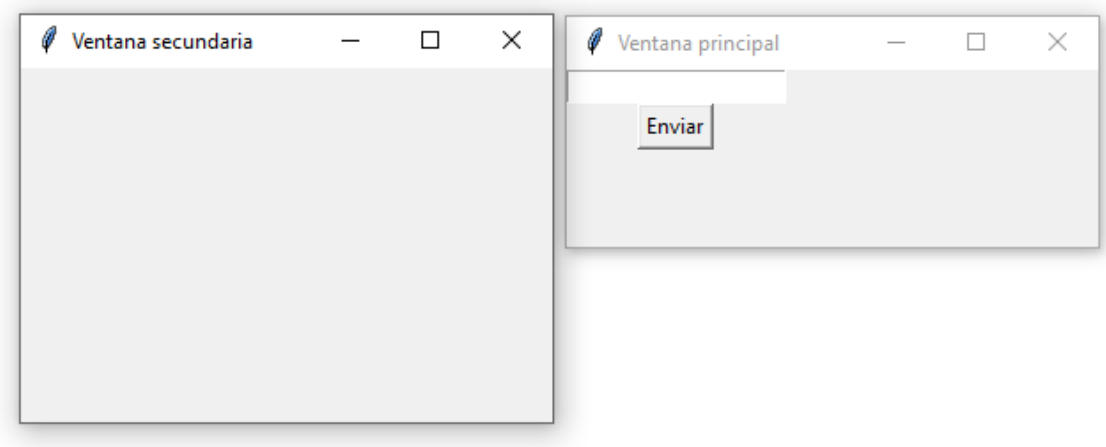
Este será el resultado:



Vamos con otro ejemplo:

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

ventana_nueva = Toplevel()
ventana_nueva.title("Ventana secundaria")
ventana_nueva.geometry("300x200")

entrada = Entry(root, width=20)
entrada.grid(row=0)

envia = Button(root, text="Enviar").grid(row=1)

mainloop()
```

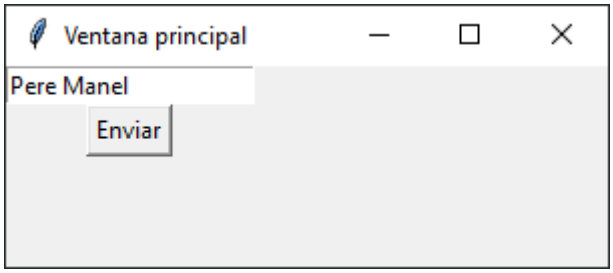
Vamos a ejecutar:



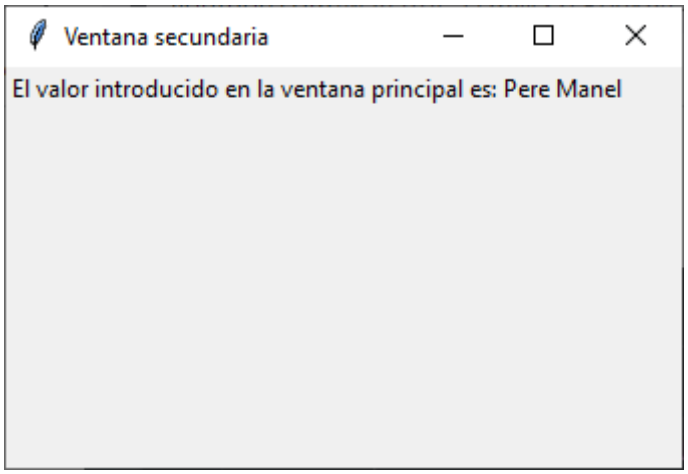
Queremos que la Ventana secundaria aparezca una vez introducido el texto y pulsado el botón enviar, es cuando se tiene que abrir la ventana secundaria con el texto.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

def enviar_boton():
    ventana_nueva = Toplevel()
    ventana_nueva.title("Ventana secundaria")
    ventana_nueva.geometry("300x200")
    valor_entrada = entrada.get()
    etiqueta = Label(ventana_nueva, text="El valor introducido en la
ventana principal es: " + valor_entrada).grid(row=0)

entrada = Entry(root, width=20)
entrada.grid(row=0)

envia = Button(root, text="Enviar", command=enviar_boton).grid(row=1)

mainloop()
```

Vamos a ejecutar:



Le damos al botón Enviar:

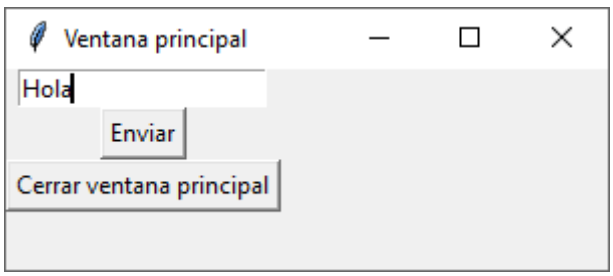


Podemos ir abriendo varias ventanas.

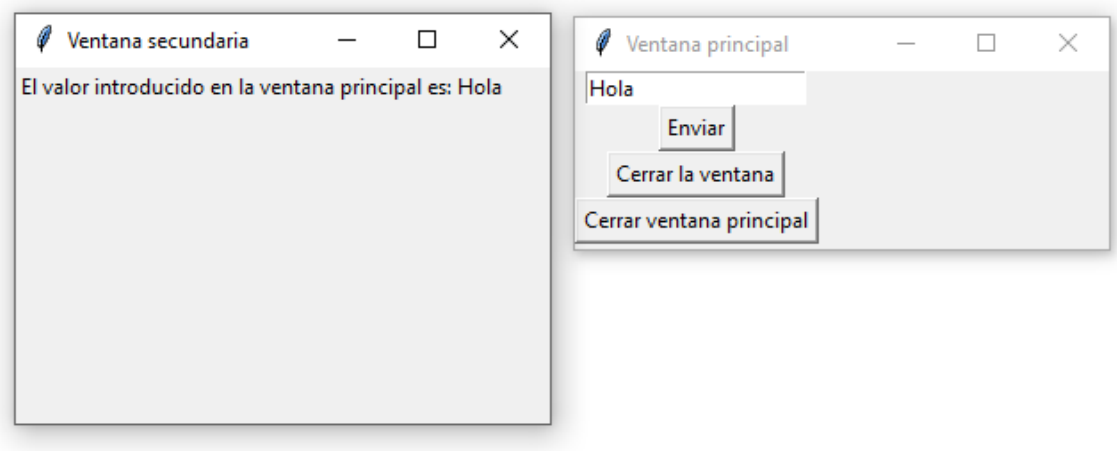
Vamos a ver como eliminar ventanas, incluso la ventana principal rood.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("300x100")

def enviar_boton():
    ventana_nueva = Toplevel()
    ventana_nueva.title("Ventana secundaria")
    ventana_nueva.geometry("300x200")
    valor_entrada = entrada.get()
    etiqueta = Label(ventana_nueva, text="El valor introducido en la
ventana principal es: " + valor_entrada).grid(row=0)
    cerrar_ventana = Button(root, text="Cerrar la ventana",
command=ventana_nueva.destroy).grid(row=2)

entrada = Entry(root, width=20)
entrada.grid(row=0)

envia = Button(root, text="Enviar", command=enviar_boton).grid(row=1)
cerrar_root = Button(root, text="Cerrar ventana principal",
command=root.destroy).grid(row=3)

mainloop()
```

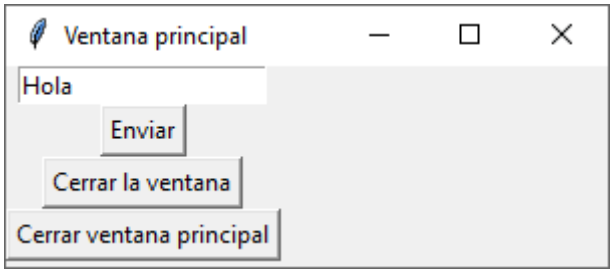
Vamos a ejecutar:



Presionamos el botón enviar:



Ahora aparece un nuevo botón en la ventana principal “Cerrar la ventana” este cerrará la ventana secundaria.
Además aparece un mensaje con la palabra que introdujimos en la ventana principal.
Vamos a darle a Cerrar la ventana.



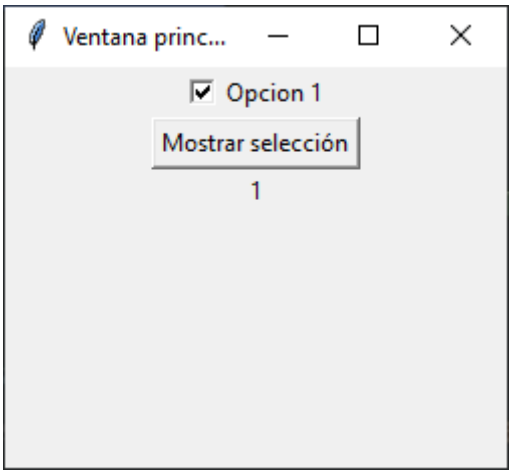
Ahora le vamos a dar a Cerrar ventana principal.

Capítulo 16: Checkbutton()

Este widget nos permite desde no coger ninguna opción o más de una opción.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("250x200")

def seleccion():
    etiqueta = Label(root, text=control.get()).pack()

control = IntVar()

opcion_1 = Checkbutton(root, text="Opcion 1", variable=control)
opcion_1.pack()

muestra_seleccion = Button(root, text="Mostrar selección",
command=seleccion).pack()


mainloop()
```

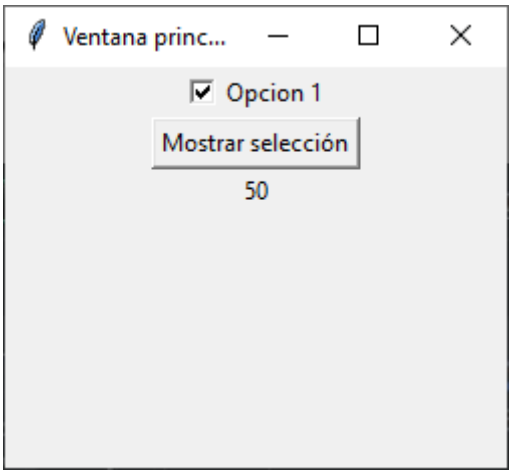
Este será el resultado:




```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("250x200")

def seleccion():
    etiqueta = Label(root, text=control.get()).pack()
```

```
control = IntVar()

opcion_1 = Checkbutton(root, text="Opcion 1", variable=control,
onvalue=50)
opcion_1.pack()

muestra_seleccion = Button(root, text="Mostrar selección",
command=seleccion).pack()


mainloop()
```

Podemos decirle que valor tiene que retornar si activamos la casilla.

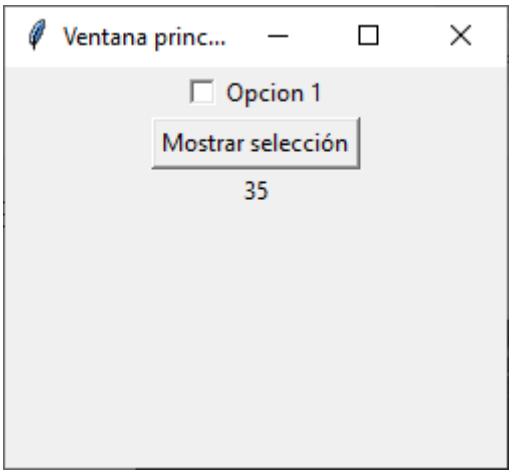



```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("250x200")

def seleccion():
    etiqueta = Label(root, text=control.get()).pack()

control = IntVar()

opcion_1 = Checkbutton(root, text="Opcion 1", variable=control,
onvalue=50, offvalue=35)
opcion_1.pack()
opcion_1.deselect()

muestra_seleccion = Button(root, text="Mostrar selección",
command=seleccion).pack()
```

```
mainloop()
```

Con la opción offvalue nos retornará el valor por defecto de 35.

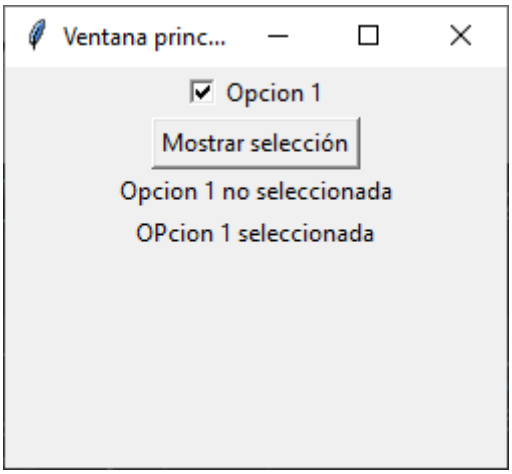
Vamos a ejecutar:



También podemos trabajar con datos String.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("250x200")

def seleccion():
    etiqueta = Label(root, text=control.get()).pack()

control = StringVar()

opcion_1 = Checkbutton(root, text="Opcion 1", variable=control,
onvalue="OPcion 1 seleccionada", offvalue="Opcion 1 no seleccionada")
opcion_1.pack()
opcion_1.deselect()

muestra_seleccion = Button(root, text="Mostrar selección",
command=seleccion).pack()


mainloop()
```

Si ejecutamos este será el resultado:

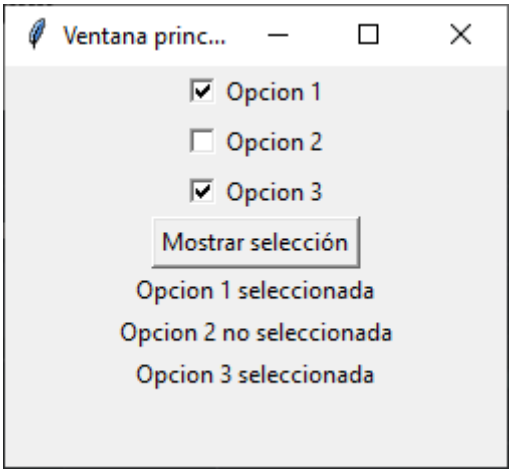


Cuando no está seleccionado muestra “Opción 1 no seleccionada” y si está seleccionado “Opción 1 seleccionada”.

```
from tkinter import *

root = Tk()
root.title("Ventana principal")
root.geometry("250x200")

def seleccion():
    etiqueta1 = Label(root, text=control1.get()).pack()
    etiqueta2 = Label(root, text=control2.get()).pack()
    etiqueta3 = Label(root, text=control3.get()).pack()

control1 = StringVar()
control2 = StringVar()
control3 = StringVar()

opcion_1 = Checkbutton(root, text="Opcion 1", variable=control1,
onvalue="Opcion 1 seleccionada", offvalue="Opcion 1 no seleccionada")
opcion_1.pack()
opcion_1.deselect()

opcion_2 = Checkbutton(root, text="Opcion 2", variable=control2,
onvalue="Opcion 2 seleccionada", offvalue="Opcion 2 no seleccionada")
opcion_2.pack()
opcion_2.deselect()

opcion_3 = Checkbutton(root, text="Opcion 3", variable=control3,
onvalue="Opcion 3 seleccionada", offvalue="Opcion 3 no seleccionada")
opcion_3.pack()
opcion_3.deselect()
```

```
muestra_seleccion = Button(root, text="Mostrar selección",
command=seleccion).pack()


mainloop()
```

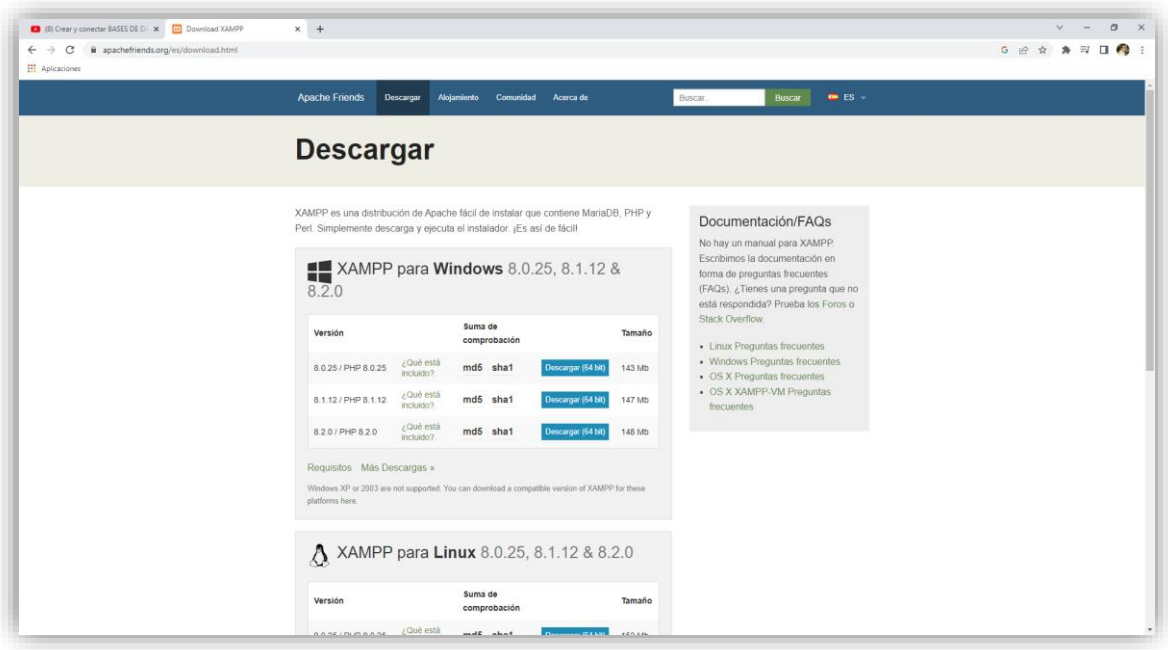
Este será el resultado:



Capítulo 17: Crear y conectar BASE DE DATOS con MariaDB

Para este proyecto tenemos que instalar XAMPP, para ello tenemos que acceder al siguiente enlace:

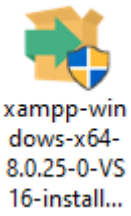
https://www.apachefriends.org/es/download.html



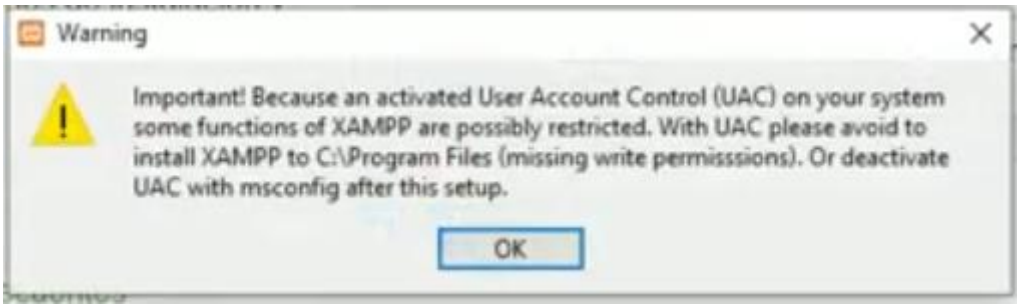
Yo me descargo la versión de Windows.

Versión		Suma de comprobación			Tamaño
8.0.25 / PHP 8.0.25	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	143 Mb
8.1.12 / PHP 8.1.12	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	147 Mb
8.2.0 / PHP 8.2.0	¿Qué está incluido?.	md5	sha1	Descargar (64 bit)	148 Mb

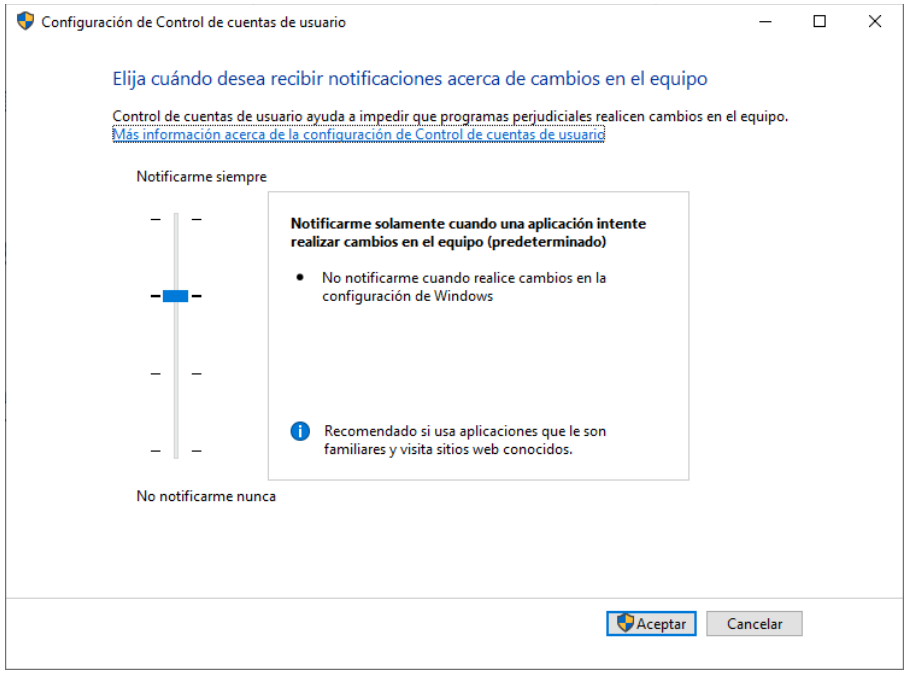
Una vez descargada nos vamos de descargas.



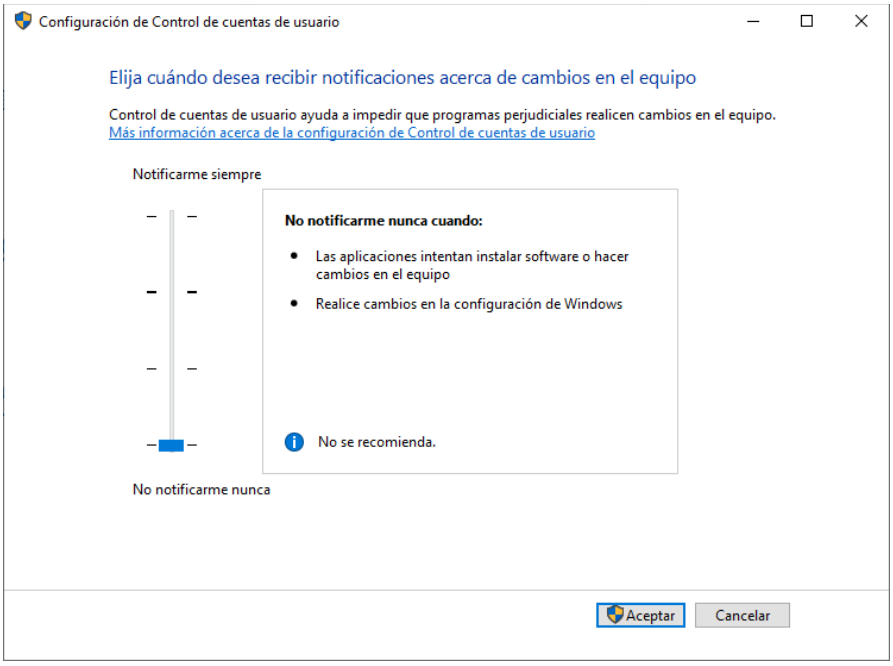
Y lo empezamos a instalar:



Si te sale este mensaje te está diciendo que puedes tener problemas con la configuración del **Control de cuentas de usuario. (En configuración)**

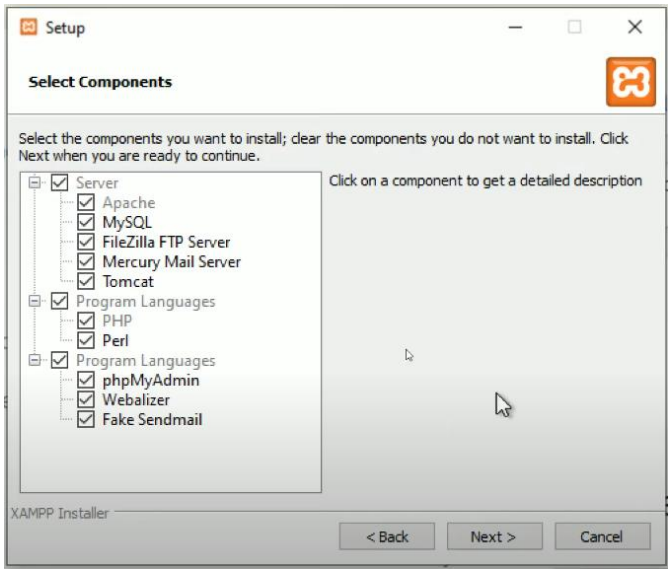


lo he cambiado:

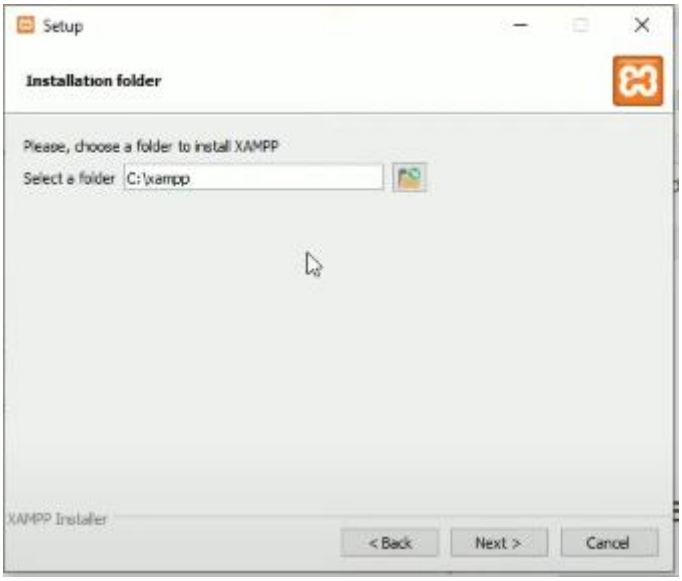


Una vez terminada la instalación déjalo como lo tenías.

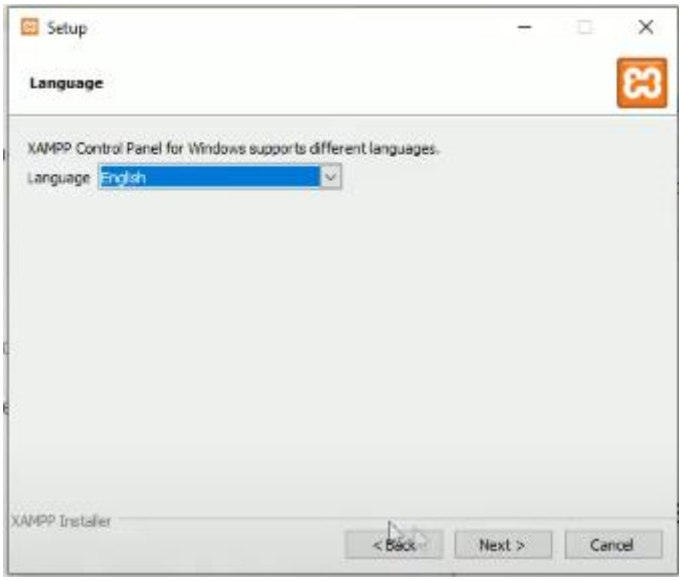
Seleccionaremos next.



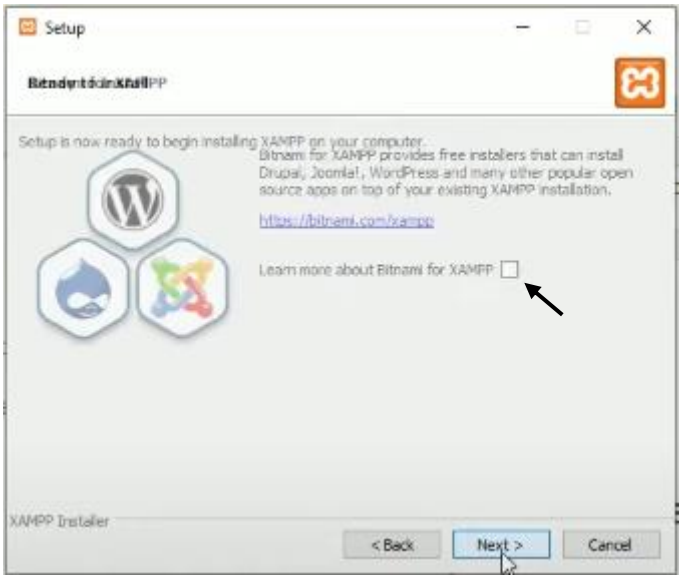
No tocaremos ninguna configuración y seleccionaremos de nuevo next.



Dejamos la ruta que pone por defecto seguido del botón next.



Lenguajes solo tiene inglés y alemán, dejaremos por defecto el inglés seguido el botón next.

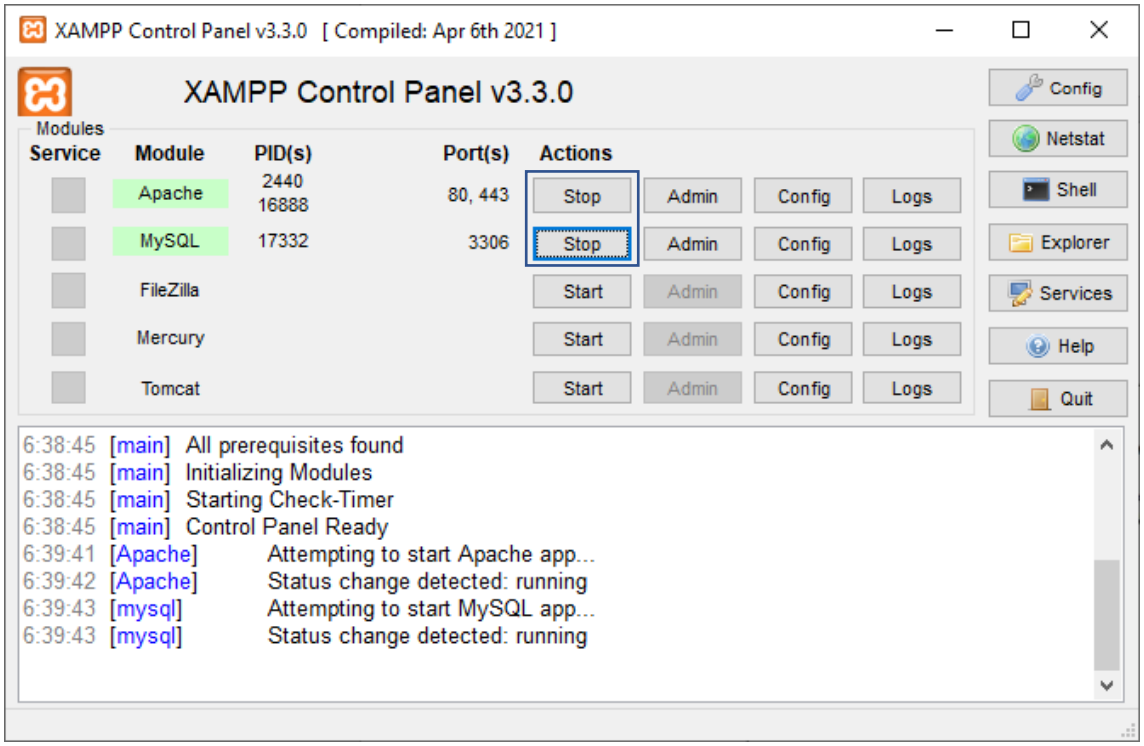


Desmarcamos la casilla seguido del botón next.

Empieza con la instalación.

Una vez finalizada la instalación:

Dejamos activa esta casilla para que se nos ejecute el programa, seguido del botón finish.

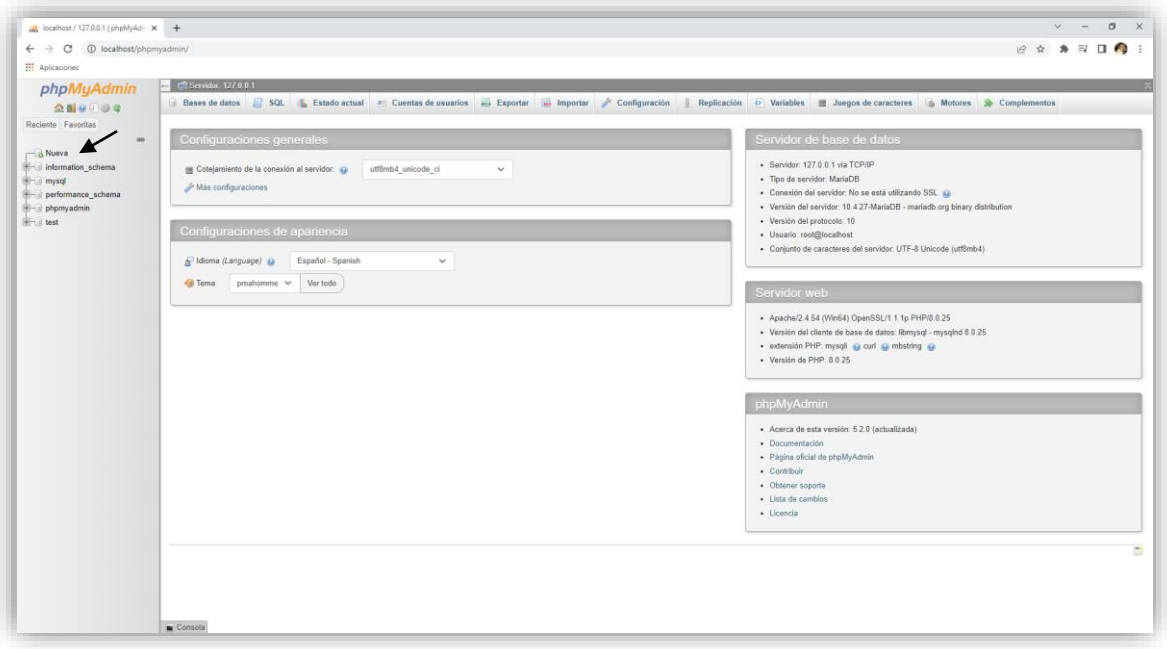


Ejecutaremos Apache y MySQL y a continuación ya podemos minimizar esta ventana.

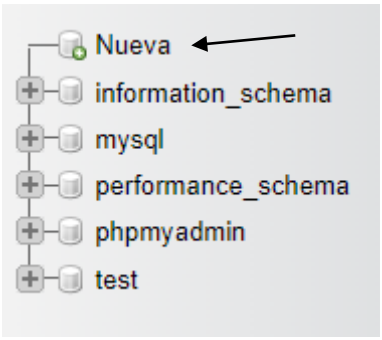
Si Apache y MySQL se muesta de color verde, significa que todo va bien.

Desde el navegador vamos a introducir la siguiente url:

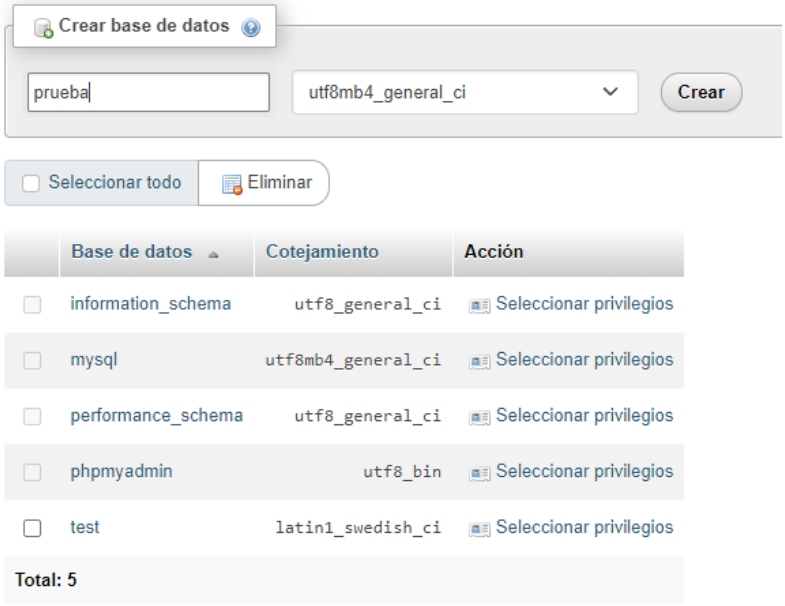
http://localhost/phpmyadmin/



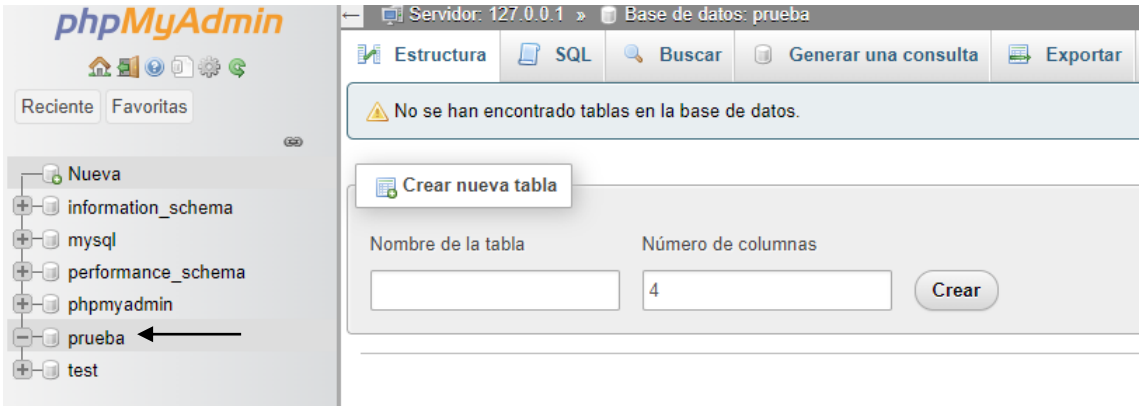
Hemos entrado al gestos de phpMyAdmin.



Vamos a seleccionar Nueva.

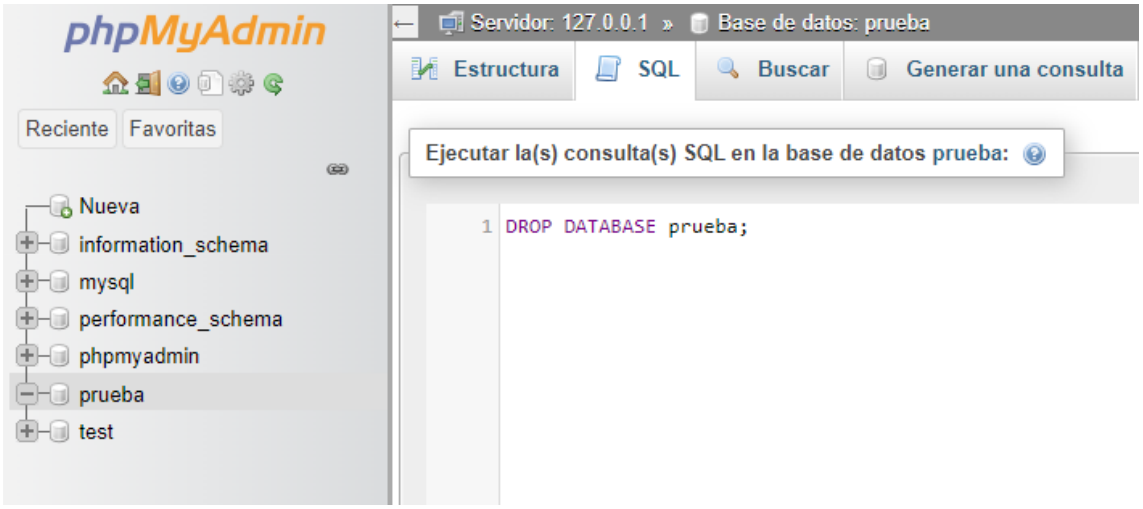


Creamos una con el nombre de prueba, seguido del botón Crear.

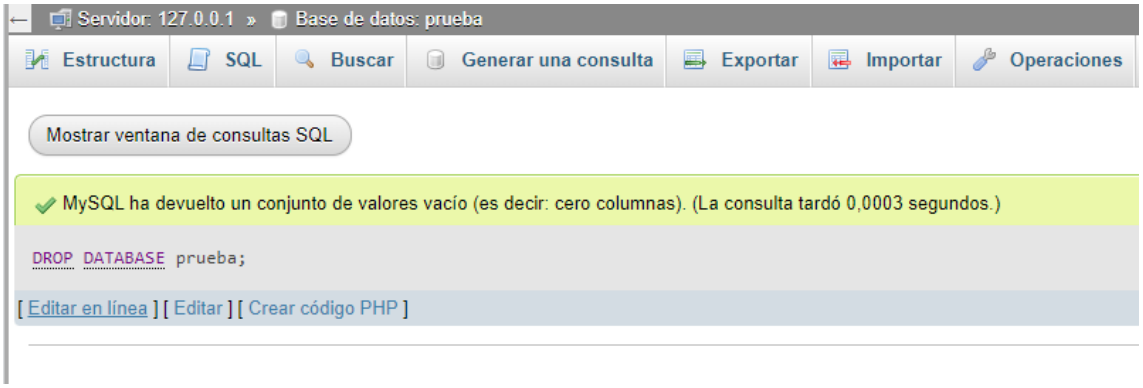


Ya está creada, pero vacía, ya es suficiente para hacer una prueba de conexión con Python.

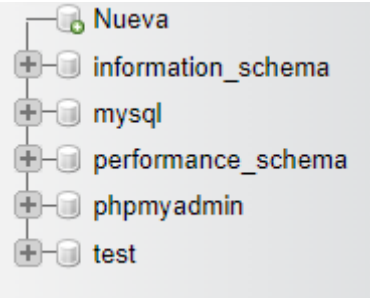
Desde la pestana SQL vamos a escribir la siguiente instrucción:



A continuación le damos al botón Continuar.

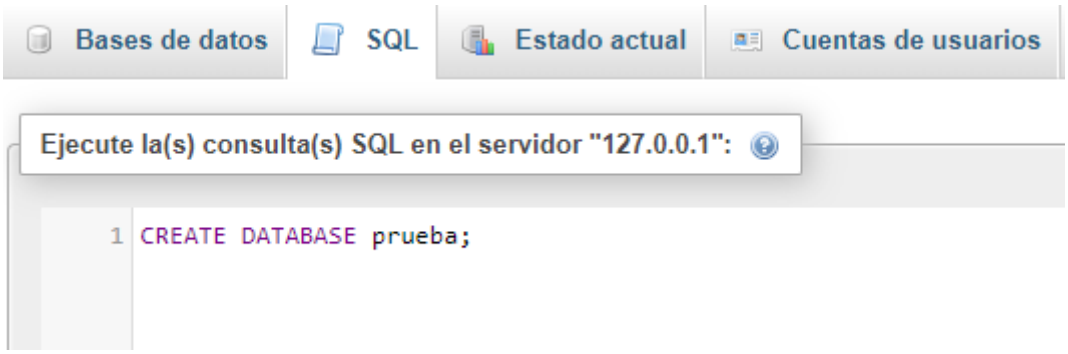


Si aparece el mensaje en verde los pasos se han realizado correctamente.

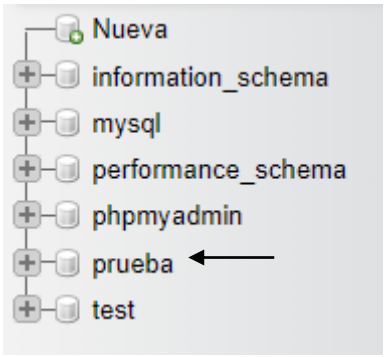


La base de datos ha sido eliminada.

Ahora vamos a crear la base de datos desde la pestaña SQL.



De nuevo le damos al botón Continuar.

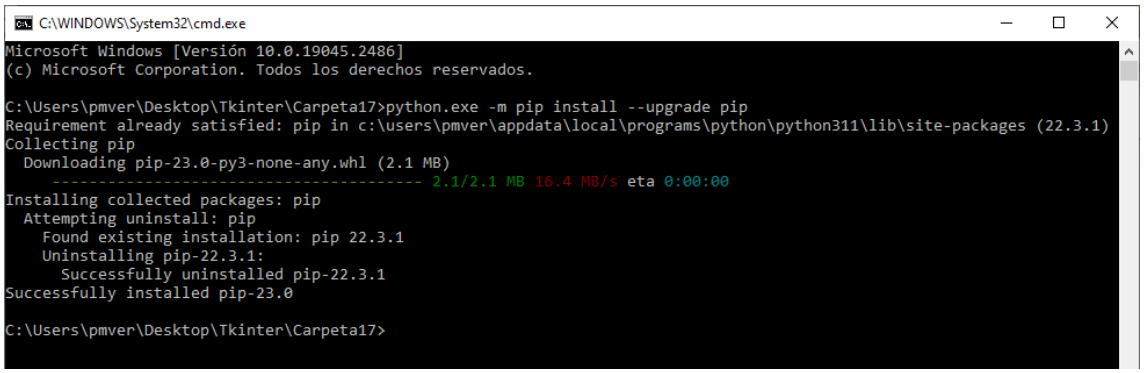


La base de datos ya está creada, esta vez por el método SQL.

Conexión a la base de datos desde Python.

Desde Visual Studio Codo en el terminal escribiremos:

Desde el cmd escribiremos. Python.exe -m pip install - -upgrade pip

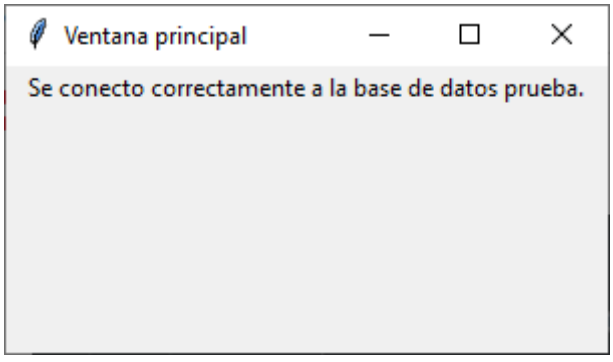


Para saber los privilegios seleccionaremos nuestra base de datos y la pestaña privilegios.

Este será el código:

```
from tkinter import *
import mariadb

root = Tk()
root.title("Ventana principal")
root.geometry("300x200")

try:
    conexion = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="prueba"
    )

    Label(root, text="Se conecto correctamente a la base de datos " +
conexion.database +".").pack()

except mariadb.Error as error:
    print(f"Error al conectar con la base de datos:{error}")

mainloop()
```

Vamos a ejecutar:

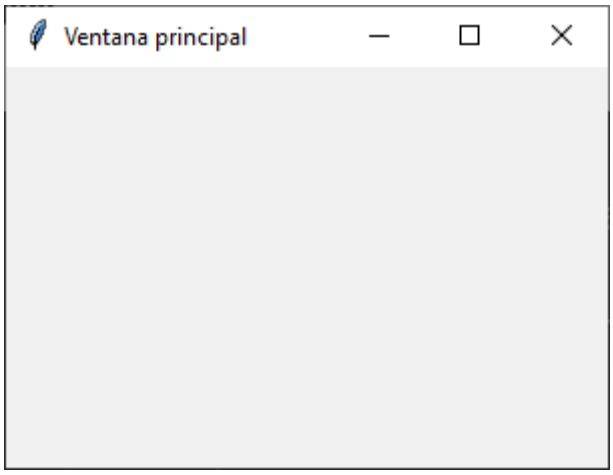


Vamos a cambiar prueba por pruebas, ya que este fichero no existe.

Ejecutamos de nuevo.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS C:\Users\pmver\Desktop\Tkinter\Carpeta17> & C:/Users/pmver/AppData/Local/Pro
grams/Python/Python311/python.exe c:/Users/pmver/Desktop/Tkinter/Carpeta17/Capi
tulo17.py
Error al conectar con la base de datos:Unknown database 'pruebas'
```

En la consola observamos el siguiente mensaje de error y la ventana se muestra:




```
conexion = mariadb.connect(
    user="roots",
    password="",
    host="127.0.0.1",
    port=3306,
    database="prueba"
)
```

Nos equivocamos con el nombre de usuario.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS C:\Users\pmver\Desktop\Tkinter\Carpeta17> & C:/Users/pmver/AppData/Local/Pro
grams/Python/Python311/python.exe c:/Users/pmver/Desktop/Tkinter/Carpeta17/Capi
tulo17.py
Error al conectar con la base de datos:Access denied for user 'roots'@'localhos
t' (using password: NO)
```

Sale el siguiente error: acceso denegado para el usuario roots.

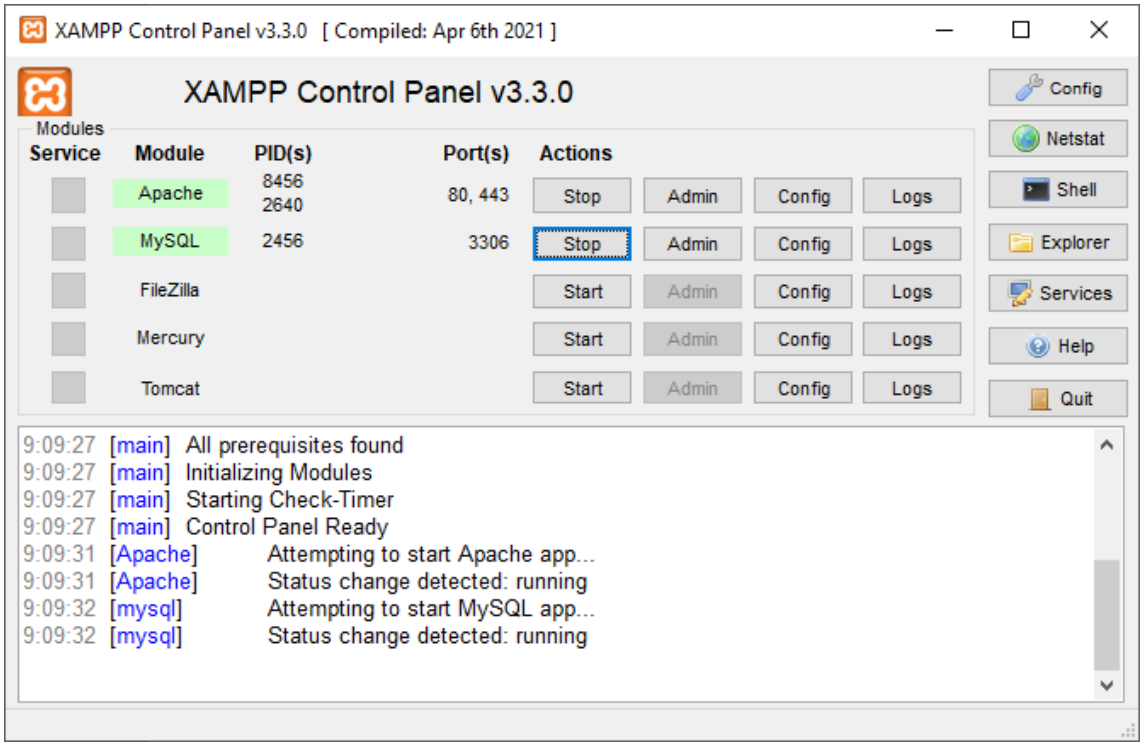
Vamos a modificar el código:

```
from tkinter import *
import mariadb
import sys
root = Tk()
root.title("Ventana principal")
root.geometry("300x200")

try:
    conexion = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="prueba"
    )

    Label(root, text="Se conecto correctamente a la base de datos " +
conexion.database +".").pack()

except mariadb.Error as error:
    print(f"Error al conectar con la base de datos:{error}")
    sys.exit(1)
mainloop()
```

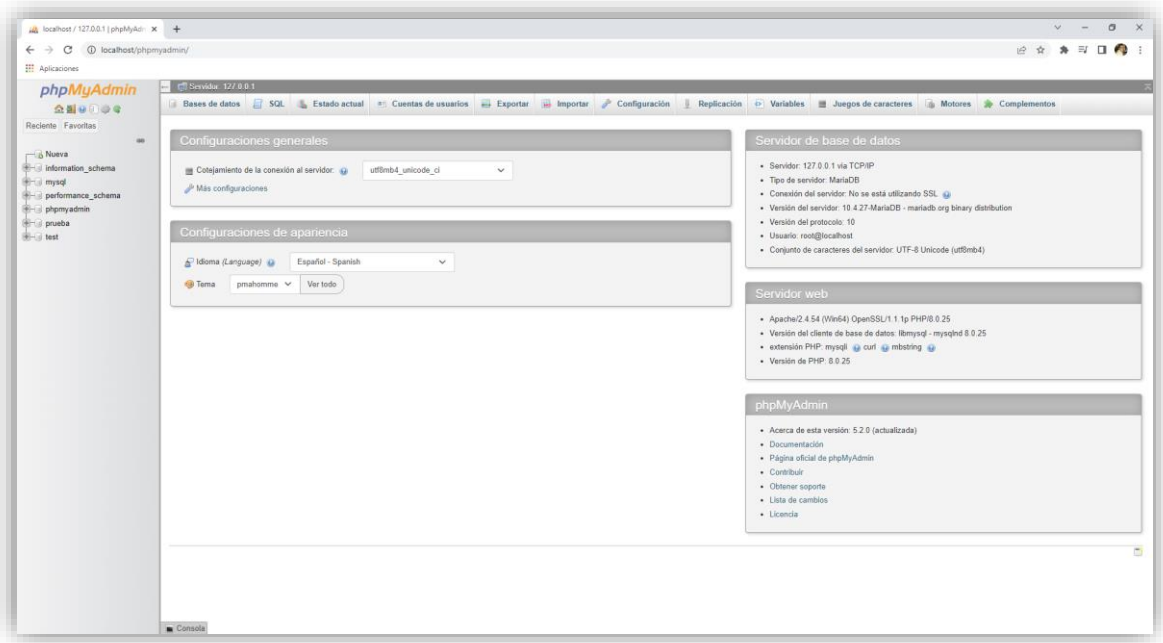
En caso que se genere un error ya no se abre la ventana y finaliza el programa.

Capítulo 18: Crear TABLAS en BASES DE DATOS con MariaDB

Vamos a seguir con el capítulo anterior, recuerda que tienes que tener en XAMPP los servicios de Apache y MySQL activados.



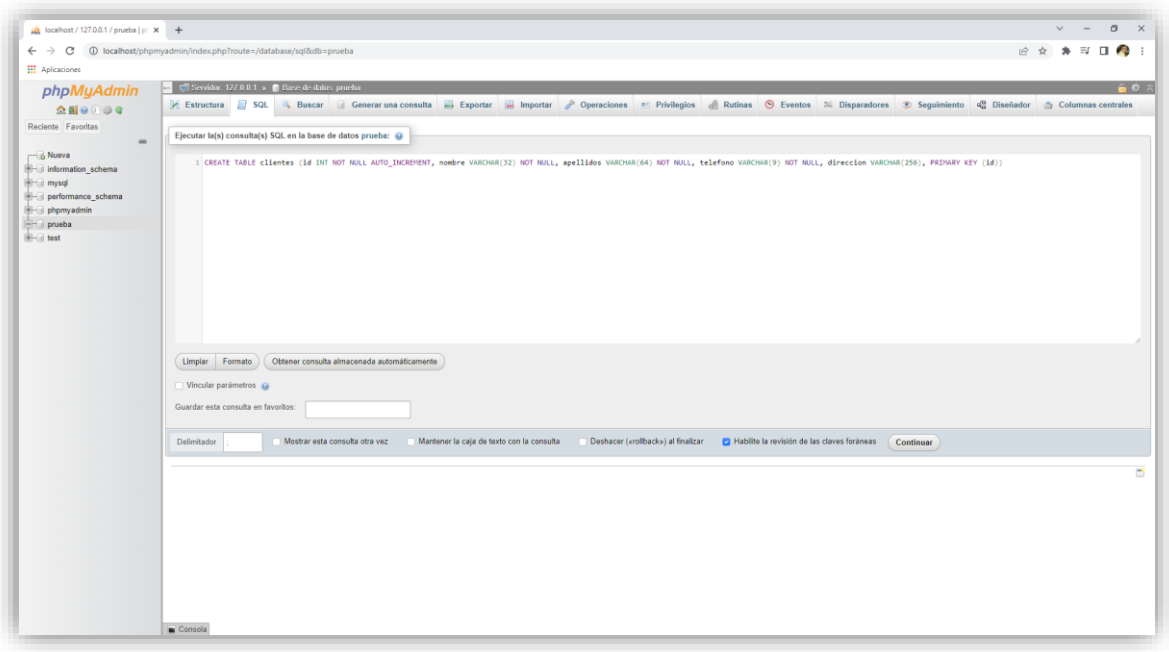
Además del navegador con la siguiente url: http://localhost/phpmyadmin/



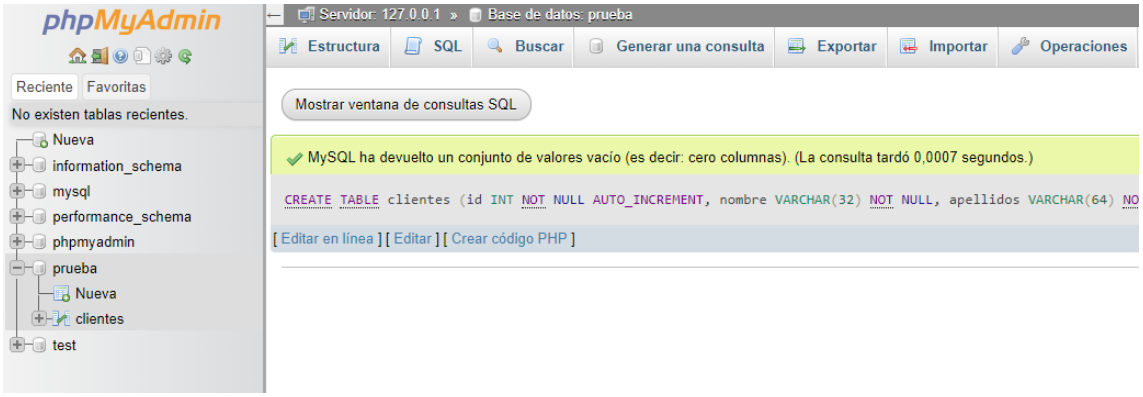
Seleccionaremos la base de datos prueba, seguido de la pestaña SQL.

Escribiremos la siguiente instrucción:

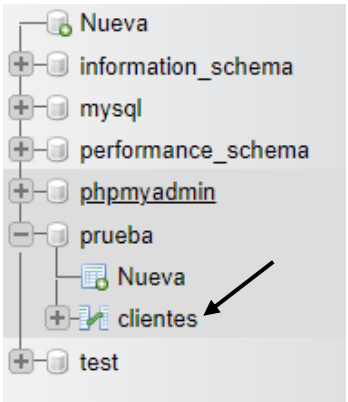
CREATE TABLE clientes (id INT NOT NULL AUTO_INCREMENT, nombre VARCHAR(32) NOT NULL, apellidos VARCHAR(64) NOT NULL, telefono VARCHAR(9) NOT NULL, direccion VARCHAR(256), PRIMARY KEY (id))



Seleccionamos el botón continuar.



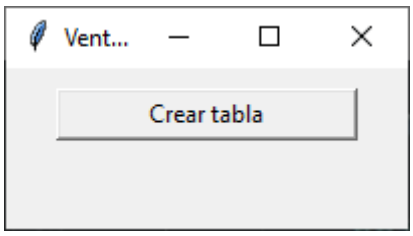
Ya se ha creado la tabla clientes:



Ahora vamos acceder a esta misma tabla de Python.

```
from tkinter import *
import mariadb
import sys
root = Tk()
root.title("Ventana principal")
root.geometry("200x80")

try:
    conexion = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="prueba"
    )
    cursor = conexion.cursor()


except mariadb.Error as error:
    print(f"Error al conectar con la base de datos:{error}")
    sys.exit(1)

def crea_tabla():
    try:
        cursor.execute("CREATE TABLE clientes (id INT NOT NULL
AUTO_INCREMENT, "
        "nombre VARCHAR(32) NOT NULL, apellidos VARCHAR(64) NOT NULL,"
        "telefono VARCHAR(9) NOT NULL, direccion VARCHAR(256), PRIMARY
KEY (id))")
        conexion.commit()
    except mariadb.Error as error_tabla:
        print(f"Error al crear la tabla: {error_tabla}")

# Interfaz gráfica
boton = Button(root, text="Crear tabla", width=20, command=crea_tabla)
boton.place(x=25, y=10)

mainloop()
```

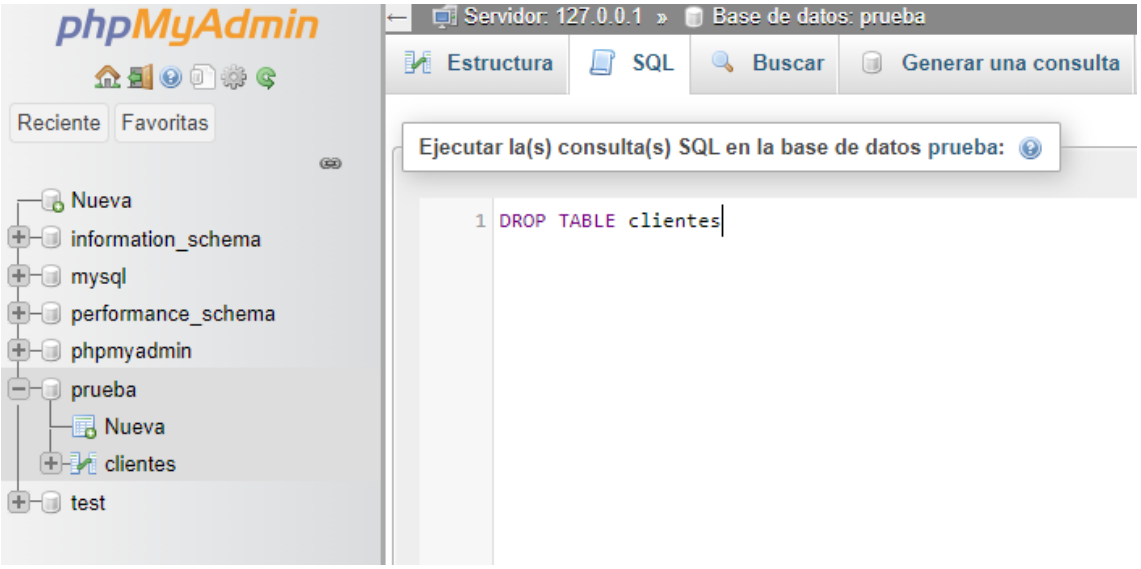
Vamos a ejecutar:



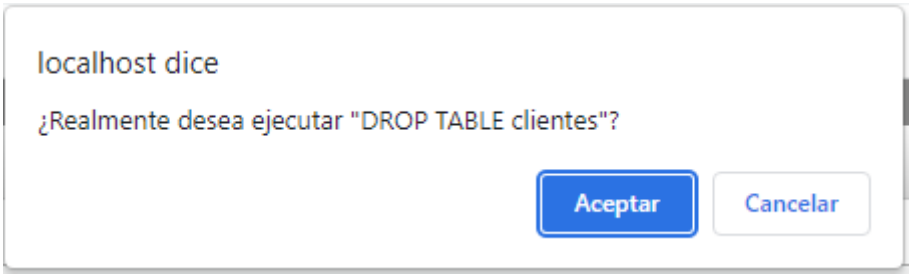
Presionamos sobre el botón “Crear tabla”.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

grams/Python/Python311/python.exe c:/Users/pmver/Desktop/Tkinter/Carpeta17/Capi
tulo17.py
Error al crear la tabla: Table 'clientes' already exists
```

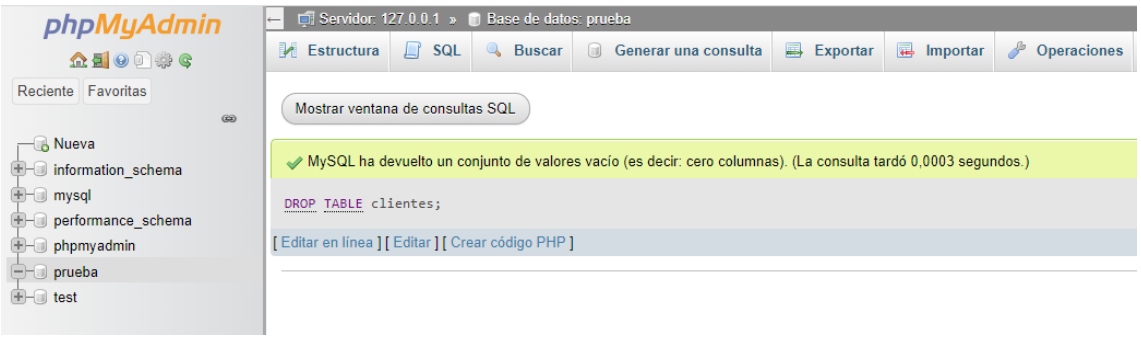
Nos dice que la tabla ya existe.



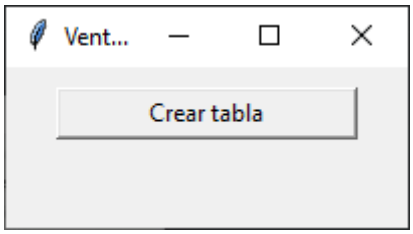
Desde SQL borramos la tabla seguido del botón continuar.



Nos pedirá confirmación, seleccionaremos Aceptar.



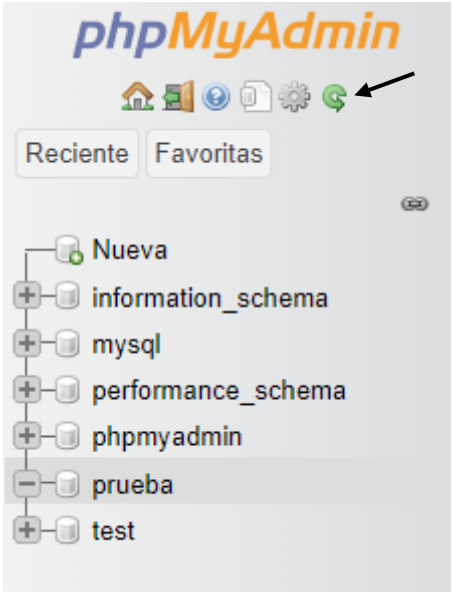
Ahora si podríamos crear la base de datos, desde visual Studio Code, ejecutaremos de nuevo.



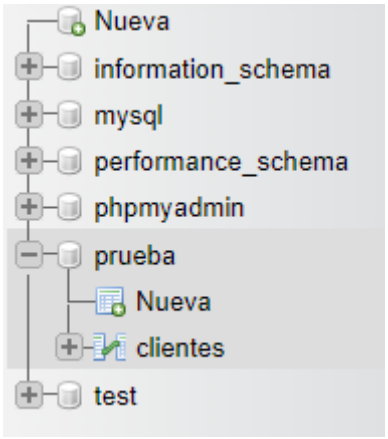
Pulsamos en “Crear tabla”.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS C:\Users\pmver\Desktop\Tkinter\Carpeta17> & C:/Users/pmver/AppData/Local/Pro
grams/Python/Python311/python.exe c:/Users/pmver/Desktop/Tkinter/Carpeta17/Capi
tulo17.py
```

Desde la consola no ha salido ningún error.



Actualizamos:



Ya hemos creado desde Python la tabla clientes.

Capítulo 19: INSERTAR datos en TABLAS – BASE DE DATOS con MariaDB

En este capítulo vamos a crear un formulario para que se puedan introducir registros.

Empezaremos con la interface gráfica:

```
from tkinter import *
import mariadb
import sys
root = Tk()
root.title("Ventana principal")
root.geometry("300x260")
```

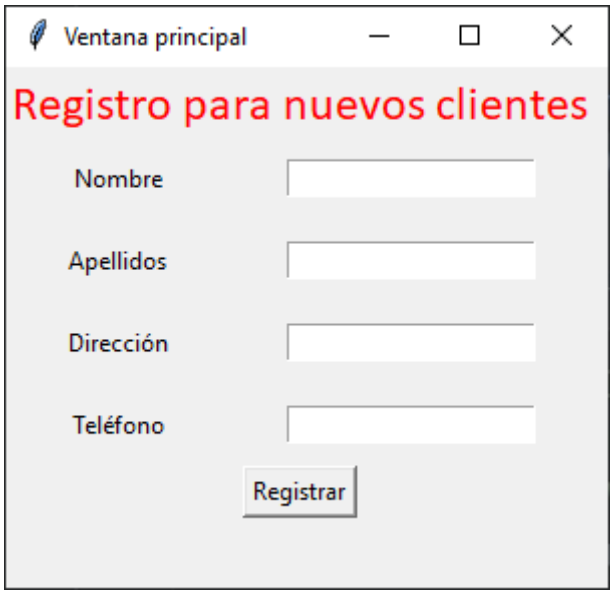
```
try:
    conexion = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="prueba"
    )
    cursor = conexion.cursor()

except mariadb.Error as error:
    print(f"Error al conectar con la base de datos:{error}")
    sys.exit(1)

def registro_cliente():
    nombre = e_nombre.get()
    apellidos = e_apellidos.get()
    telefono = e_telefono.get()
    direccion = e_direccion.get()
    try:
        cursor.execute("INSERT INTO clientes(nombre, apellidos, telefono,
direccion)VALUES(?,?,?,?)",(nombre, apellidos, telefono, direccion))
        conexion.commit()  # Para que guarde los cambios en la base de
datos
    except mariadb.Error as error_registro:
        print(f"Error en el registro: {error_registro}")
```

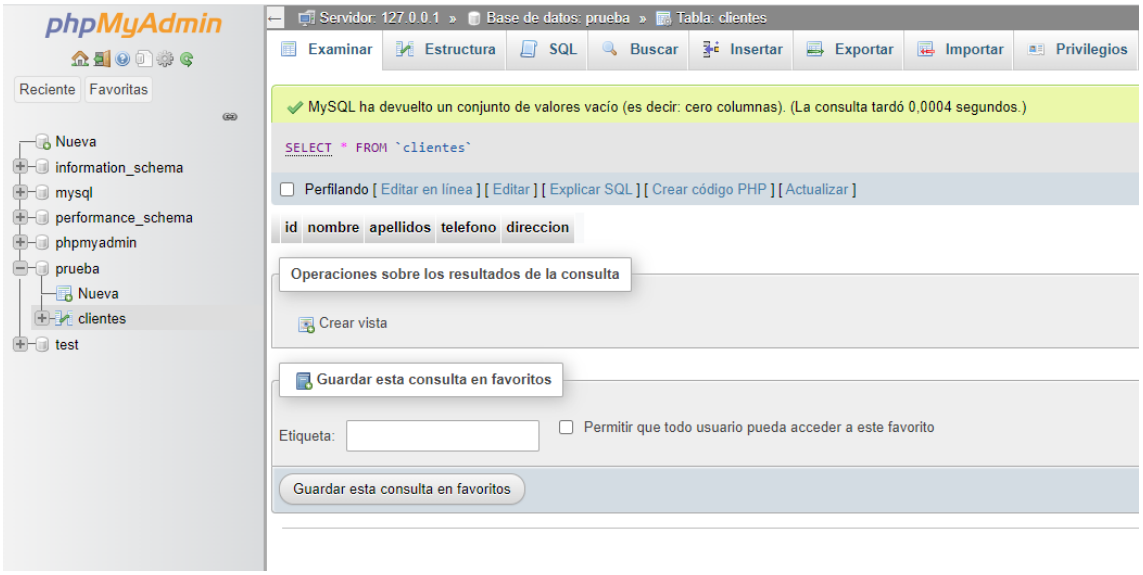
```
# Interfaz gráfica
Label(root,
      text="Registro para nuevos clientes",
      font="calibri 18",
      fg="red").grid(row=0, columnspan=2)
```

```
Label(root,
      text="Nombre").grid(row=1, column=0, pady=10)

e_nombre = Entry(root)
e_nombre.grid(row=1, column=1)

Label(root,
      text="Apellidos").grid(row=2, column=0, pady=10)

e_apellidos = Entry(root)
e_apellidos.grid(row=2, column=1)

Label(root,
      text="Dirección").grid(row=4, column=0, pady=10)

e_direccion = Entry(root)
e_direccion.grid(row=4, column=1)

Label(root,
      text="Teléfono").grid(row=5, column=0, pady=10)

e_telefono = Entry(root)
e_telefono.grid(row=5, column=1)

boton = Button(root,
        text="Registrar",
        command=registro_cliente).grid(row=6, columnspan=2)

mainloop()
```

La parte que está fuera del marco en la parte gráfica y lo enmarcado en la parte lógica.

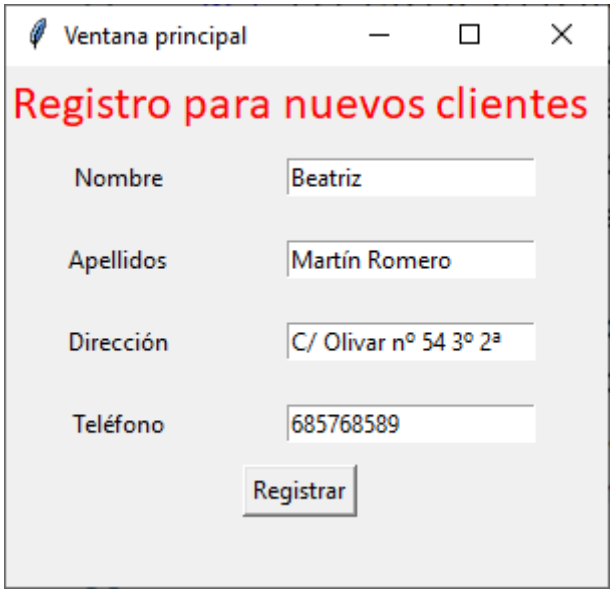
Vamos a ejecutar:





La base de datos con la tabla vacía.

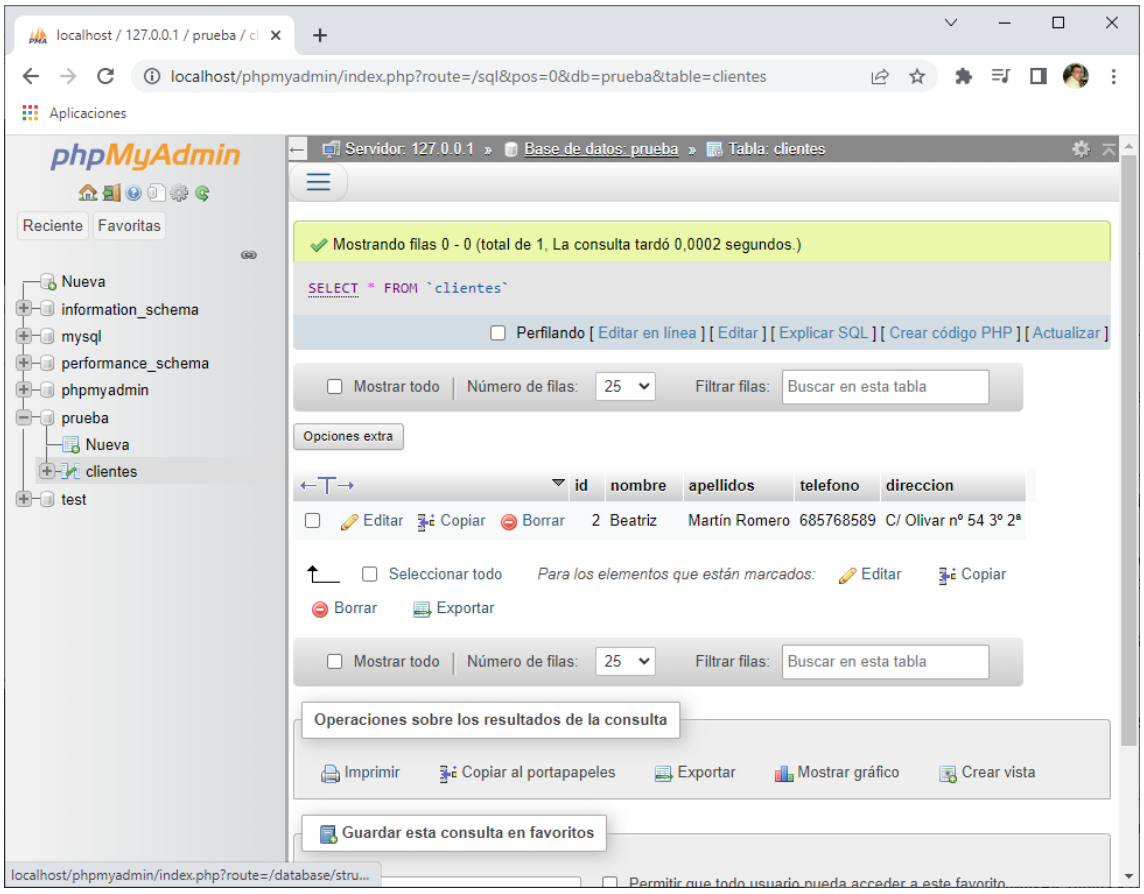
Vamos a ejecutar el programa para agregar un registro:

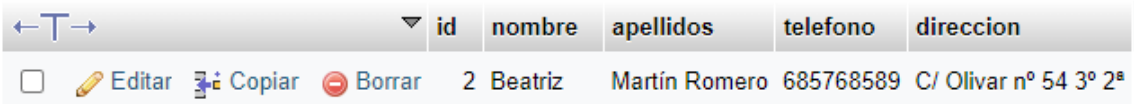


A continuación de daremos al botón Registrar.

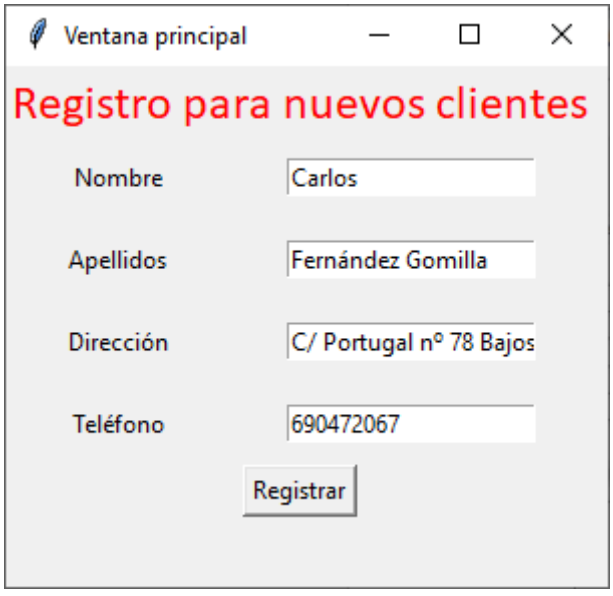
En la consola no habido ningún mensaje de error.

Ahora vamos a consultar la base de datos:





Vamos a introducir un segundo cliente:



Lo registramos.

	id	nombre	apellidos	telefono	direccion
Editar Copiar Borrar	2	Beatriz	Martín Romero	685768589	C/ Olivar nº 54 3º 2ª
Editar Copiar Borrar	3	Carlos	Fernández Gomilla	690472067	C/ Portugal nº 78 Bajos

Si queremos que una vez añadido el registro se borren todos los campos así tiene que ser la función:

```
def registro_cliente():
    nombre = e_nombre.get()
    apellidos = e_apellidos.get()
    direccion = e_direccion.get()
    telefono = e_telefono.get()
    e_nombre.delete(0, END)
    e_apellidos.delete(0, END)
    e_direccion.delete(0, END)
    e_telefono.delete(0, END)
    try:
        cursor.execute("INSERT INTO clientes (nombre, apellidos,
telefono, direccion)VALUES(?,?,?,?)",(nombre, apellidos, telefono,
direccion))
        conexion.commit()  # Para que guarde los cambios en la base de
datos
    except mariadb.Error as error_registro:
        print(f"Error en el registro: {error_registro}")
```

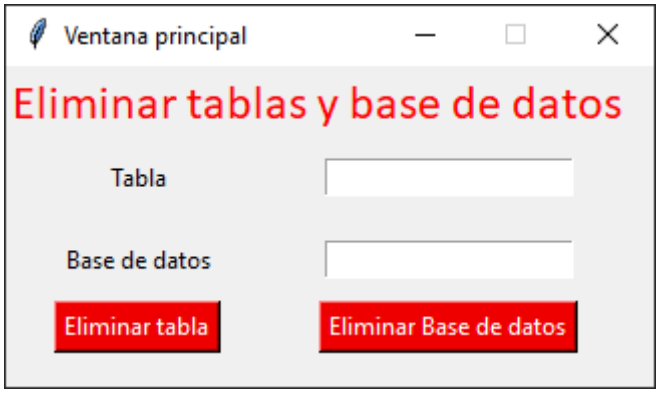
Capítulo 20: ELIMINAR Tablas y Base de datos con MariaDB

Este último proyecto vamos a realizar un programa que va a ser capaz de eliminar una tabla o una base de datos, escribiendo solo sus nombres.

```
from tkinter import *
import mariadb
import sys

root = Tk()
root.title("Ventana principal")
root.geometry("323x160")
root.resizable(0,0)

# Conexión a la base de datos

try:
    conexion = mariadb.connect(
        user="root",
        password="",
        host="127.0.0.1",
        port=3306,
        database="prueba"
    )
    # Obtiene el cursor
    cursor = conexion.cursor()
except mariadb.Error as error:
    print(f"Error al conectar con la base de datos {error}")
    sys.exit(1)


    # Interface gráfica

Label(root,
    text="Eliminar tablas y base de datos",
    font="calibri 18",
    fg="red").grid(row=0, columnspan=2)

Label(root,
    text="Tabla").grid(row=1, column=0, pady=10)

e_tabla = Entry(root)
e_tabla.grid(row=1, column=1, pady=10)

Label(root,
    text="Base de datos").grid(row=2, column=0, pady=10)

e_base_datos = Entry(root)
e_base_datos.grid(row=2, column=1, pady=10)
```

```
boton1 = Button(root,
        text="Eliminar tabla",
        bg="red2",
        fg="white").grid(row=5, column=0)

boton2 = Button(root,
        text="Eliminar Base de datos",
        bg="red2",
        fg="white").grid(row=5, column=1)

mainloop()
```

Si ejecutamos veremos el entorno gráfico:



Vamos a crear la función de eliminar Tabla.

```
def eliminar_tabla():
    tabla = e_tabla.get()
    try:
        cursor.execute(f"DROP TABLE {tabla}")
        conexion.commit()
    except mariadb.Error as error_tabla:
        print(f"Error al elimiar la tabla: {error_tabla}")
```

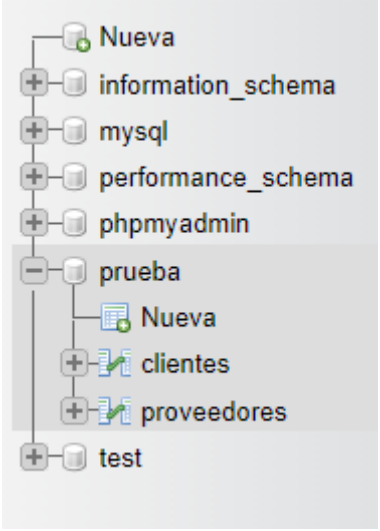
Vamos a crear la función eliminar Base de datos:

```
def eliminar_base_datos():
    base_datos = e_base_datos.get()
    try:
        cursor.execute(f"DROP DATABASE {base_datos}")
        conexion.commit()
    except mariadb.Error as error_bd:
        print(f"Error al elimiar la Base de datos: {error_bd}")
```

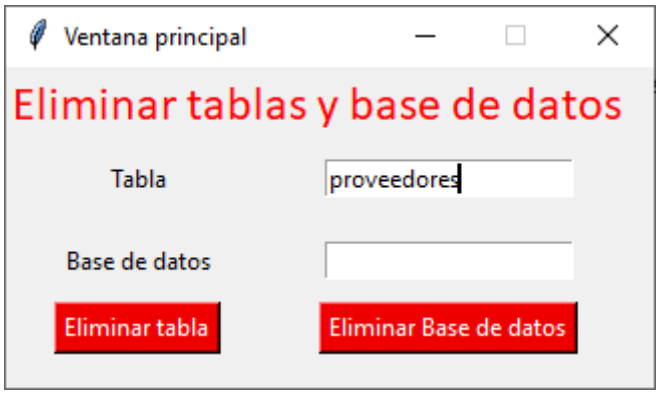
Vamos a añadir a los Button el correspondiente command.

```
boton1 = Button(root,
         text="Eliminar tabla",
         bg="red2",
         fg="white",
         command=eliminar_tabla).grid(row=5, column=0)

boton2 = Button(root,
         text="Eliminar Base de datos",
         bg="red2",
         fg="white",
         command=eliminar_base_datos).grid(row=5, column=1)
```

Hemos creado la tabla proveedores:

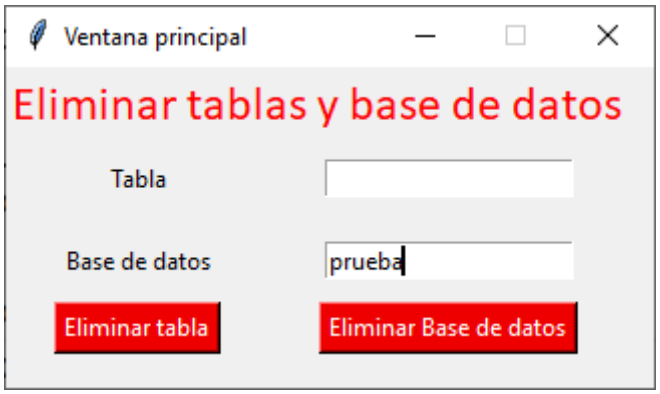


Lo vamos a ejecutar para eliminar la tabla proveedores:



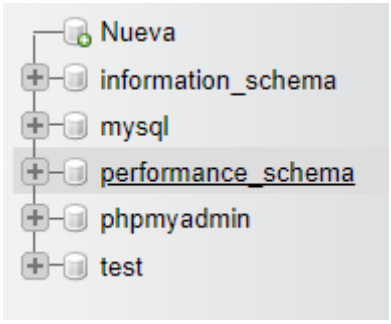
Presionamos el botón Eliminar tabla.

Ahora vamos a eliminar la base de datos.



Presionaremos el botón Eliminar Base de datos.

Ya hemos eliminado la base de datos:



Contenido