Módulo Tkinter.

Introducción.

El módulo Tkinter, incluido en la instalación de Python, nos permite crear una interfaz gráfica para nuestros programas.
Cuenta con una serie de elementos gráficos llamados Widgets fácilmente configurables gracias a los cuales armaremos nuestra interfaz.
Estos elementos los iremos colocando en base a una jerarquía dentro de nuestra aplicación, ciertos elementos solo pueden usarse dentro de otros.
En esta unidad vamos a ver los Widgets principales, pero lo que aprendamos será mas que suficiente para podamos revisar la documentación de este módulo y hacer nuestras interfaces tan complejas como sea necesario.

Elemento TK.

Es el contenedor principal de todos los demás elementos, es la raíz de nuestra interfaz. No tiene un tamaño propio definido, se va adaptando a los elementos que le coloquemos en su interior.
Lo primero que debemos hacer para trabajar con este módulo, es importarlo:

```
from tkinter import *
```

Con esto, tenemos disponibles todos los métodos y funcionalidades de la librería para utilizar en nuestro programa.

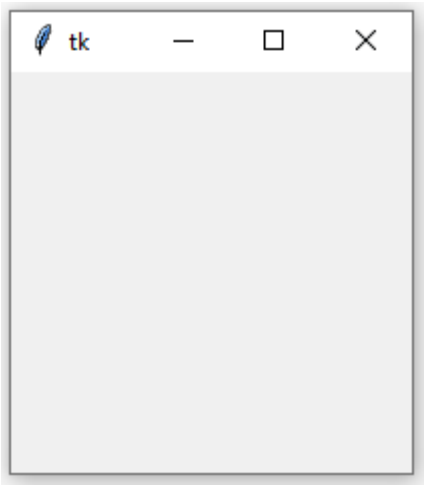
El segundo paso es crear nuestro contenedor principal, la raíz de nuestra interfaz, y para ello le tenemos que dar un nombre, por ejemplo, "ventana".

```
from tkinter import *

ventana = Tk()
```

A continuación, creamos un bucle dentro del cual estará contenido nuestro programa:

```
from tkinter import *

ventana = Tk()
ventana.mainloop()
```

Si guardamos los cambios y ejecutamos nuestro código, veremos nuestra ventana creada, simple, sencilla y por el momento sin ninguna configuración.



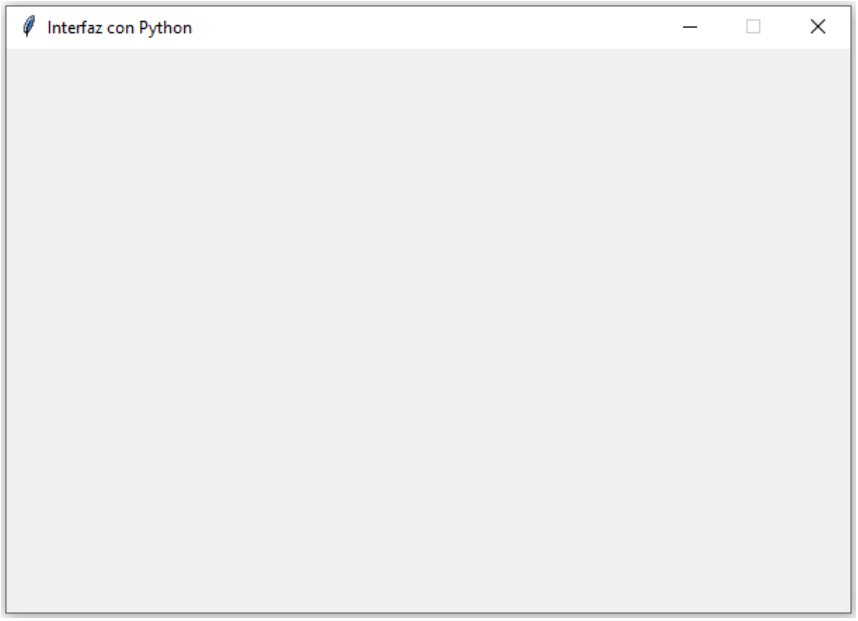
Veamos algunas de las configuraciones que le podemos hacer a nuestra ventana:

- Poner título: title("Título de la ventana")
- Desactivar la redimensión de la ventana: resizable(0,0)
- Asignar tamaño: geometry(ancho x alto)

Veamos como aplicar estos métodos en nuestro código:

```
from tkinter import *

ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("600x400")
ventana.mainloop()
```

Si guardamos y ejecutamos, veremos la siguiente interfaz, con la opción de agrandar la ventana, grisada.



Etiquetas.

Para poner etiquetas dentro de nuestras ventanas, utilizamos la clase LABEL.

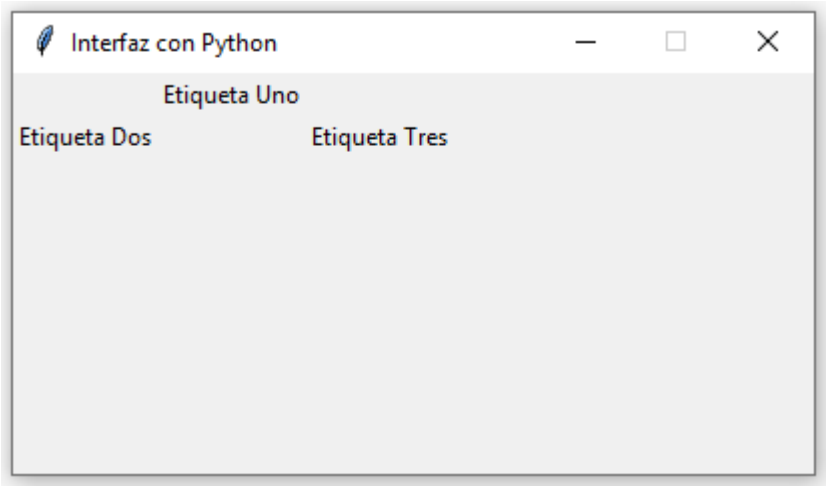
Le asignamos un nombre a nuestra etiqueta, especificamos a que contenedor pertenece, en nuestro caso, como estamos trabajando con la raíz como contenedor, será el nombre de nuestra raíz el que le pasaremos (ventana) y finalmente el texto que queremos mostrar.

A continuación, tenemos que decirle a Python en qué lugar de nuestra ventana queremos que se vea utilizando el método GRID de la clase Label. Los parámetros dicho método son ROW y COLUMN.

Para entender cómo funciona GRID, vamos a colocar tres etiquetas diferentes.

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

# Etiquetas
etiqueta_uno = Label(ventana,text='Etiqueta Uno')
etiqueta_uno.grid(row=0,column=1)

etiqueta_dos = Label(ventana,text='Etiqueta Dos')
etiqueta_dos.grid(row=1,column=0)

etiqueta_tres = Label(ventana,text='Etiqueta Tres')
etiqueta_tres.grid(row=1,column=3)

ventana.mainloop()
```

Al método GRID no tenemos que pasarle las coordenadas exactas en pixeles, solo la posición relativa dentro de la ventana, a la que dividiremos en líneas y columnas.
Si ejecutamos el ejemplo anterior, veremos una ventana así:



La primera etiqueta (etiqueta1) la ubicamos en la primera fila (fila cero) y la segunda columna (columna 1).

La segunda etiqueta (etiqueta2), está ubicada en la segunda fila (fila uno) y la primera columna (columna cero).
Por último, la tercera etiqueta (etiqueta3) la colocamos en la misma fila que la etiqueta anterior, pero en la segunda columna (columna uno).

Básicamente lo que hacemos es dividir nuestra ventana, en filas y columnas:

row =1 , column=1	row =1 , column=2	row =1 , column=3	row =1 , column=4	row =1 , column=5
row =2 , column=1	row =2 , column=2	row =2 , column=3	row =2 , column=4	row =2 , column=5
row =3 , column=1	row =3 , column=2	row =3 , column=3	row =3 , column=4	row =3 , column=5
row =4 , column=1	row =4 , column=2	row =4 , column=3	row =4 , column=4	row =4 , column=5

****** ¿Qué sucede si a dos etiquetas diferentes les damos las mismas coordenadas?***

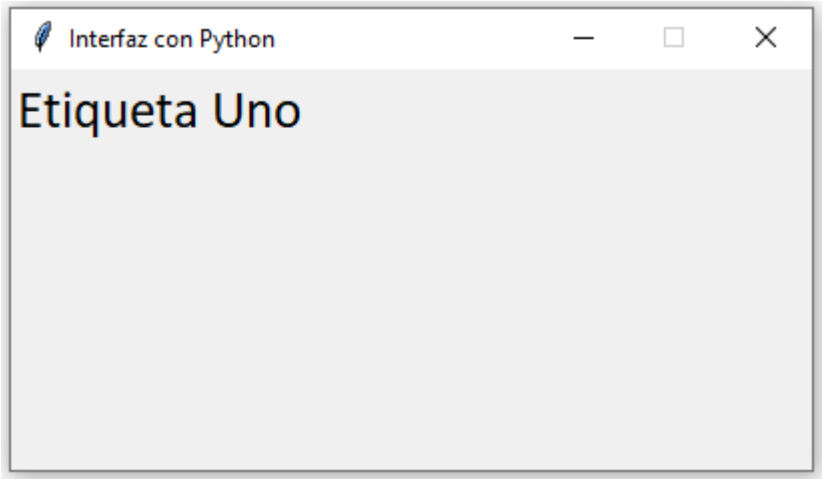
****** ¿Qué pasa si intentamos ubicar una etiqueta en una fila o columna determinada dejando sobre la etiqueta y a su derecha filas y columnas en blanco?***

Fuente y color de las etiquetas.

Podemos darles a nuestras etiquetas un tipo específico de fuente, tamaño y color, tanto de texto como color de fondo.
Veamos como elegir la fuente y el tamaño de la misma, dentro de la clase LABEL.

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

# Etiquetas
etiqueta_uno = Label(ventana,text='Etiqueta Uno',font=("calibri",20))
etiqueta_uno.grid(row=0,column=2)

ventana.mainloop()
```

Así se ve nuestra interfaz, aprovechamos el ejemplo para poner la etiqueta en la columna 2 para ver como reacciona Python en un caso así, donde queremos poner un elemento en una columna, dejando columnas libres a su izquierda.



Como vemos, a pesar de pedir que la etiqueta está en la columna 2, al no haber nada en las columnas cero y uno, el elemento queda alineado a la izquierda.

Para el color, utilizamos los atributos bg(color de fondo) y fg(color de fuente).
Por ejemplo, queremos una etiqueta de fondo negro y color de texto rojo:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

# Etiquetas
etiqueta_uno = Label(ventana,text='Etiqueta Uno',font=("calibri",20),bg='black',fg='red')
etiqueta_uno.grid(row=0,column=2)

ventana.mainloop()
```

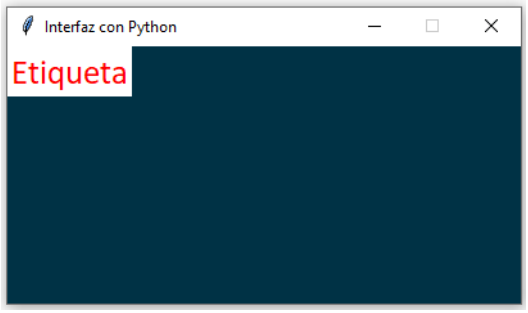
Los valores para el color también podemos pasarlos en hexadecimal:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

# Etiquetas
etiqueta_uno = Label(ventana,text='Etiqueta',font=("calibri",20),bg='#ffffff',fg='red')
etiqueta_uno.grid(row=0,column=2)

ventana.mainloop()
```

Método configure.

Con este método de nuestro objeto ventana, podemos configurar algunos aspectos de vista y funcionamiento, por ahora vemos cómo cambiar el color de fondo:

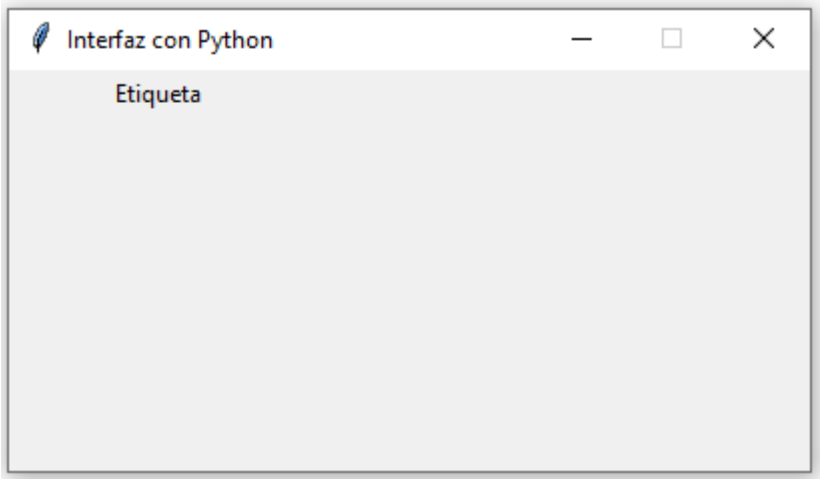
```
ventana.configure(background='#003245')
```



Método PLACE.

Como vimos, el método GRID es muy útil si queremos ubicar varios elementos dentro de nuestra ventana (en este caso etiquetas) sin necesidad de andar calculando coordenas de X e Y, pero no es muy cómodo si lo que queremos es colocar nuestro elemento exactamente en una coordenada.
Veamos cómo podemos usarlo para colocar nuestra etiqueta, por ejemplo, en las coordenas x=50 e y=1:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")


# Etiquetas
etiqueta_uno = Label(ventana,text='Etiqueta')
etiqueta_uno.place(x=50,y=1)

ventana.mainloop()
```

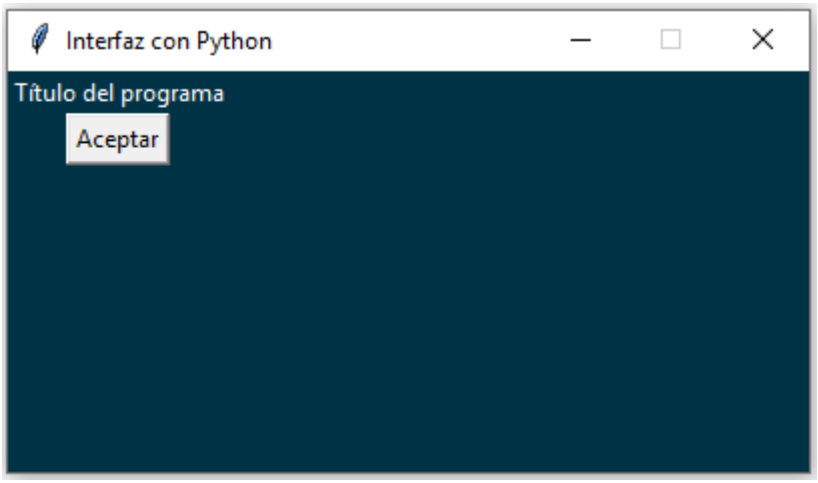


De acuerdo a nuestra necesidad, utilizaremos uno u otro método.
GRID cuando tenemos varios elementos para ubicar y queremos que se adapten automáticamente al tamaño de la ventana o PLACE cuando necesitemos colocar un elemento en una coordenada específica.

Botones.

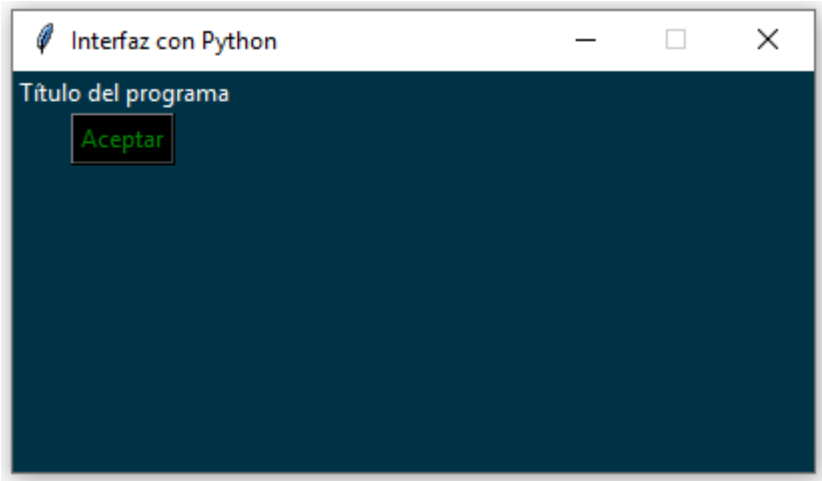
Para crear y usar botones, procedemos de forma muy similar a como lo hicimos con las etiquetas.

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")
ventana.configure(background='#003245')


# Etiquetas
etiqueta_uno = Label(ventana,text='Título del programa',bg='#003245',fg='#ffffff')
etiqueta_uno.grid(column=1,row=0)

boton_uno = Button(ventana,text='Aceptar')
boton_uno.grid(column=1,row=1)

ventana.mainloop()
```



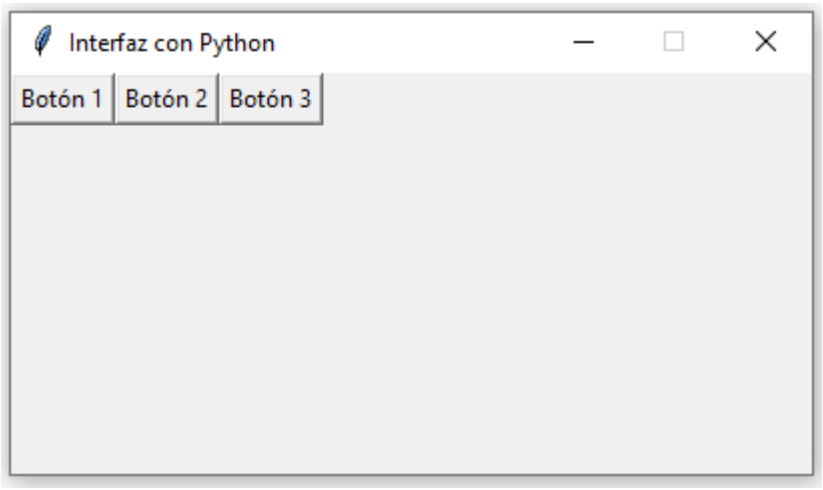
Podemos cambiar los atributos visuales de la misma forma que lo hicimos con las etiquetas, por ejemplo, queremos un botón de fondo negro con letras verdes:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")
ventana.configure(background='#003245')


# Etiquetas
etiqueta_uno = Label(ventana,text='Título del programa',bg='#003245',fg='#ffffff')
etiqueta_uno.grid(column=1,row=0)

boton_uno = Button(ventana,text='Aceptar',bg='black',fg='green')
boton_uno.grid(column=1,row=1)

ventana.mainloop()
```



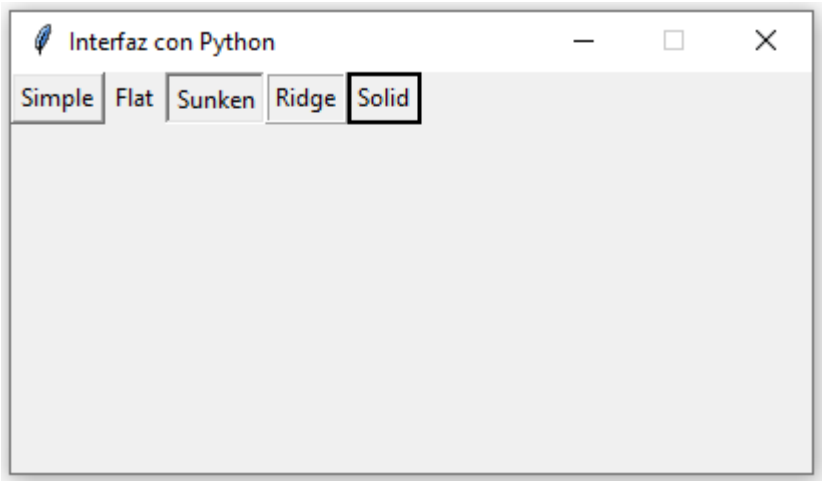
Si necesitamos más de un botón, deberemos crearlos de la misma manera, dando un nombre diferente a cada uno, y aprovechando el método GRID los acomodaremos muy fácilmente:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

boton_uno = Button(ventana,text='Botón 1')
boton_uno.grid(column=1,row=1)

boton_dos = Button(ventana,text='Botón 2')
boton_dos.grid(column=2,row=1)

boton_tres = Button(ventana,text='Botón 3')
boton_tres.grid(column=3,row=1)

ventana.mainloop()
```



RELIEVES: los botones tienen una propiedad para el relieve, la forma en que se verán por pantalla.
Esta propiedad es RELIEF y puede tomar los valores FLAT, SUNKEN, RIDGE o SOLID.
Veamos un ejemplo de cada uno:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

boton_1 = Button(ventana,text='Simple')
boton_1.grid(column=1,row=1)

boton_2 = Button(ventana,text='Flat',relief=FLAT)
boton_2.grid(column=2,row=1)

boton_3 = Button(ventana,text='Sunken',relief=SUNKEN)
boton_3.grid(column=3,row=1)

boton_4 = Button(ventana,text='Ridge',relief=RIDGE)
boton_4.grid(column=4,row=1)

boton_5 = Button(ventana,text='Solid',relief=SOLID)
boton_5.grid(column=5,row=1)

ventana.mainloop()
```

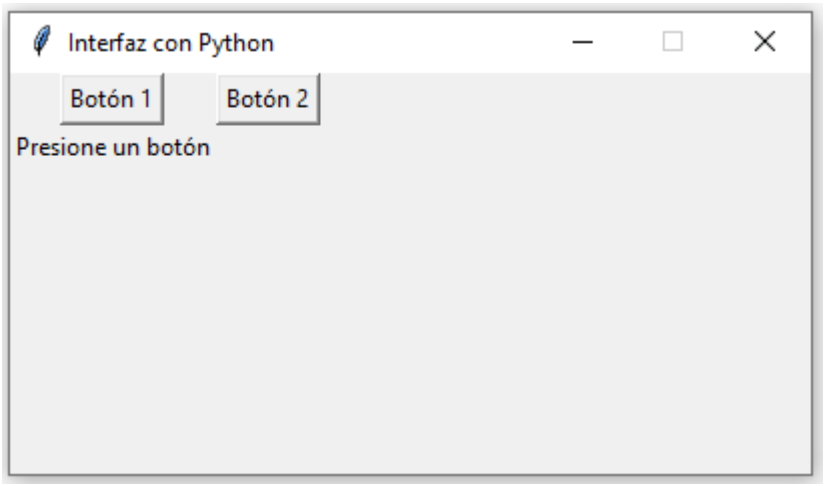


Atributo COMMAND.

Este atributo del botón nos permite definir el código a ejecutar cada vez que el botón sea presionado.

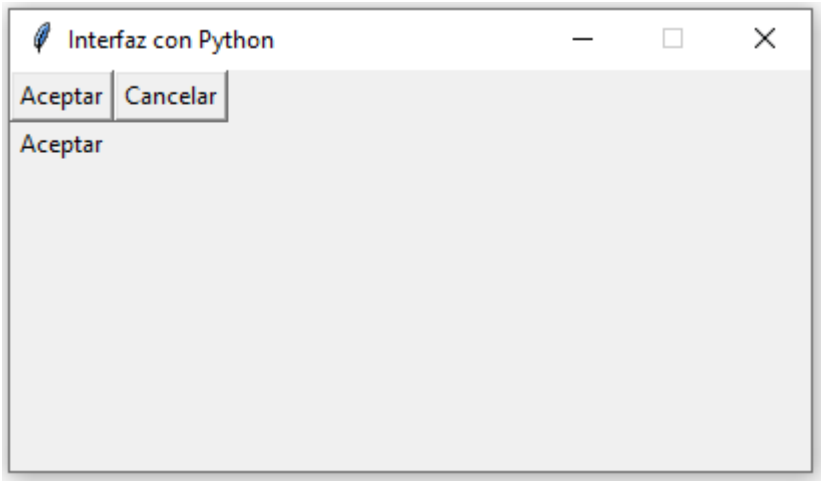
Veamos un ejemplo con dos botones cada uno asociado a una función diferente:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

def click_uno():
        lbl_uno.configure(text='Se presiono el botón 1')

def click_dos():
        lbl_uno.configure(text='Se presiono el botón 2')

boton_1 = Button(ventana,text='Botón 1',command=click_uno)
boton_1.grid(column=0,row=0)

boton_2 = Button(ventana,text='Botón 2',command=click_dos)
boton_2.grid(column=1,row=0)

lbl_uno = Label(ventana,text='Presione un botón')
lbl_uno.grid(column=0,row=1)

ventana.mainloop()
```



Lo que hace el con este código es definir dos funciones a las que llamamos click1 y click2 y en cada uno de los botones, mediante el atributo COMMAND definimos que función será invocada con cada uno.
En este ejemplo podemos apreciar un poco mejor el funcionamiento de GRID, que acomodó el tamaño de la primera columna al elemento más ancho, contenido en dicha columna, que en nuestro caso es la etiqueta.
Probemos modificar la etiqueta y el texto de cada botón:

```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

def click_uno():
        lbl_uno.configure(text='Aceptar')

def click_dos():
        lbl_uno.configure(text='Cancelar')

boton_1 = Button(ventana,text='Aceptar',command=click_uno)
boton_1.grid(column=0,row=0)

boton_2 = Button(ventana,text='Cancelar',command=click_dos)
boton_2.grid(column=1,row=0)

lbl_uno = Label(ventana,text='')
lbl_uno.grid(column=0,row=1)

ventana.mainloop()
```



Ahora el elemento más ancho de cada columna es el botón, y ya no se va a modificar la distribución de los elementos al presionar cada uno:

Ingreso de datos.

Para ingresar datos desde el teclado, usamos la clase Entry, de manera similar a como creamos las etiquetas y los botones, indicándole a que ventana pertenece y mediante GRID, la colocamos en nuestro programa.

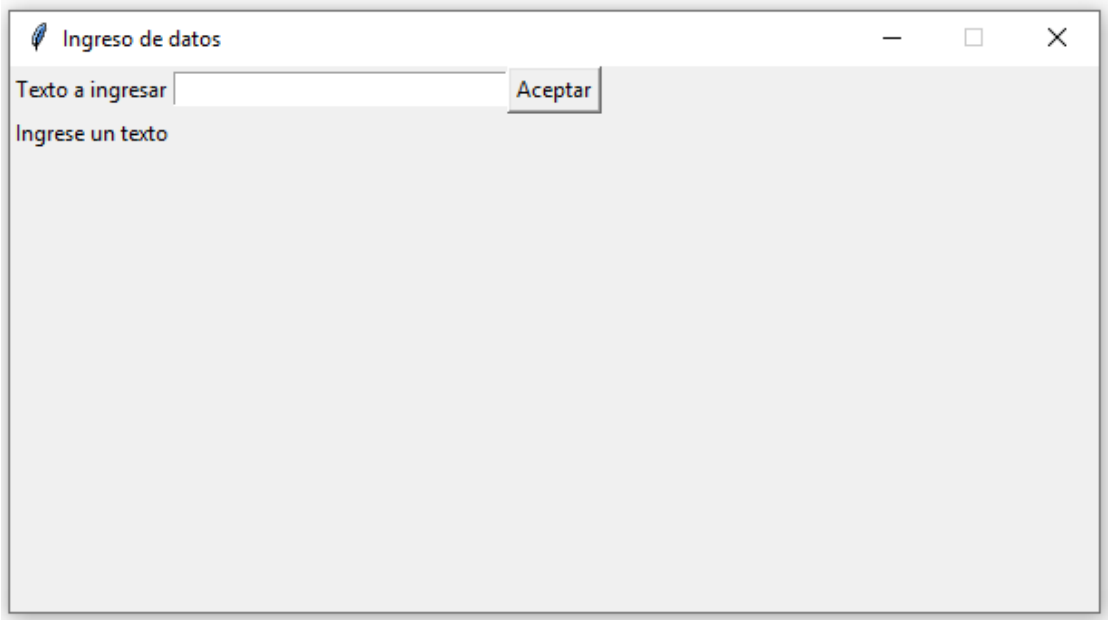
```
from tkinter import *
ventana = Tk()
ventana.title("Interfaz con Python")
ventana.resizable(0,0)
ventana.geometry("400x200")

caja = Entry(ventana)
caja.grid(row=0,column=0)

ventana.mainloop()
```

El valor que ingresemos por teclado será guardado dentro del elemento llamado caja.
Veamos como recuperar y utilizar dicho valor haciendo un ejemplo práctico de un programa que nos permita escribir una palabra, y luego, presionando un botón, nos muestre dicha palabra por pantalla.

```
from tkinter import *
ventana = Tk()
ventana.title("Ingreso de datos")
ventana.resizable(0,0)
ventana.geometry("600x300")

def click():
        mensaje=texto.get()
        etiqueta_dos.config(text=mensaje)

etiqueta_uno=Label(ventana,text='Texto a ingresar')
etiqueta_uno.grid(row=0,column=0)

etiqueta_dos=Label(ventana,text='Ingrese un texto')
etiqueta_dos.grid(row=1,column=0)

texto=Entry(ventana,width=30)
texto.grid(row=0,column=1)

boton=Button(ventana,text='Aceptar',command=click)
boton.grid(row=0,column=2)

ventana.mainloop()
```

Lo que hicimos fue, en primer lugar, definir la función que cambia el valor de la etiqueta dos por el valor que ingresemos por teclado mediante ENTRY.

Luego creamos las etiquetas, la caja de texto para que el usuario ingrese una frase y finalmente el botón que invoca a la función.



Cada vez que ingresamos una frase en la caja de texto y presionamos el botón "Aceptar", este invoca a la función que le cambia el valor por defecto a la etiqueta dos por el valor ingresado en la caja de texto.

Las cajas de texto pueden deshabilitarse mediante un atributo de estado.

```
texto=Entry(ventana,width=30,state='disable')
texto.grid(row=0,column=1)
```

Para borrar el contenido de una caja de texto, utilizamos el método DELETE que recibe como parámetros el inicio y el final, en nuestro caso, al ser una caja de 30 caracteres de ancho, le pasamos los valores cero y 30 como inicio y final.

Mensajes.

Para mostrarle mensajes emergentes al usuario, vamos a utilizar el módulo MESSAGEBOX.

Veamos un ejemplo simple de como funciona, haciendo un programa que muestre dos botones y que, al presionarlo, salga un mensaje emergente diferente para cada uno.

```
from tkinter import *
from tkinter import messagebox as MessageBox

ventana = Tk()
ventana.title("Ingreso de datos")
ventana.resizable(0,0)
ventana.geometry("400x200")

def texto_a():
    MessageBox.showinfo("Mensaje", "Excelente")

def texto_b():
    MessageBox.showinfo("Mensaje", "No sabes nada de fútbol")

btn1=Button(ventana, text = "Soy de Chaca", command=texto_a)
btn1.grid(row=0,column=0)
btn2=Button(ventana, text = "Soy de otro club", command=texto_b)
btn2.grid(row=1,column=0)

ventana.mainloop()
```