Interfaces gráficas en Python con Tkinter

1. Introducción

Tkinter es el paquete más utilizado para crear interfaces gráficas en Python. Es una capa orientada a objetos basada en Tcl (sencillo y versátil lenguaje de programación open-source) y Tk (la herramienta GUI estándar para Tcl).

El paquete Tkinter está incluido en Python como un paquete estándar, por lo que no es necesario instalar nada para usarlo.

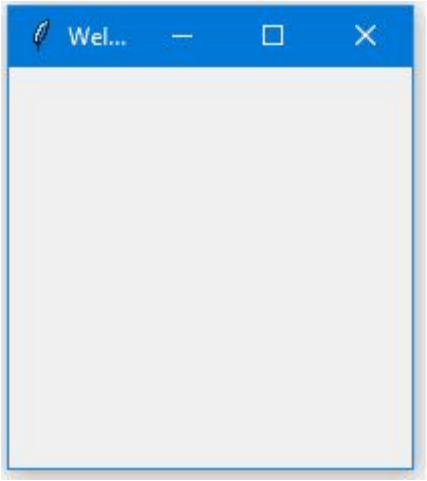
El paquete Tkinter es un paquete muy poderoso. Si ya ha instalado Python, puede usar IDLE, que es el IDE integrado que se envía con Python

Crear tu primera aplicación GUI

Primero, importamos el paquete Tkinter y crearemos una ventana y estableceremos su título:

```
from tkinter import *

window = Tk()

window.title("Bienvenido a mi App")

window.mainloop()
```

El resultado debe verse como esto:



¡Increíble! Nuestra aplicación funciona.

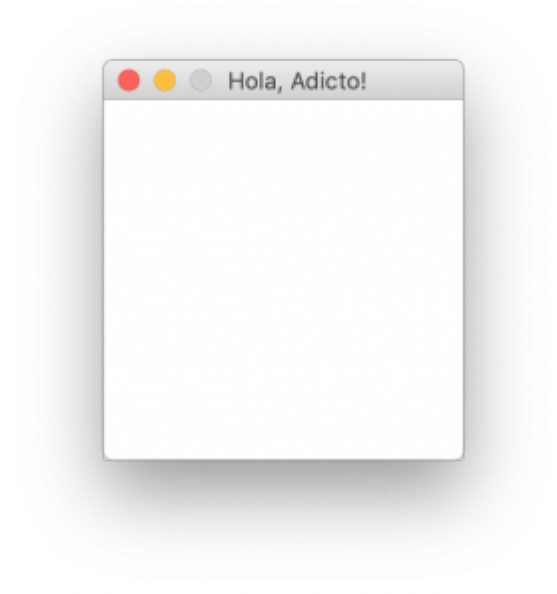
La última línea llama a la función `mainloop`. Esta función llama al ciclo sin fin de la ventana, por lo que la ventana esperará cualquier interacción del usuario hasta que la cerremos.

Si olvidas llamar a la función `mainloop`, no aparecerá nada al usuario.

2. Widgets

A la hora de montar una vista con Tkinter, nos basaremos en widgets jerarquizados, que irán componiendo poco a poco nuestra interfaz. Algunos de los más comunes son:

Tk: es la raíz de la interfaz, donde vamos a colocar el resto de widgets.



Agregar una etiqueta a nuestro ejemplo anterior, crearemos

una etiqueta usando la clase `label` de la siguiente manera:

```
lbl = Label(window, text="Hello")
```

Luego estableceremos su posición en el formulario utilizando la función `grid` con su ubicación, de esta manera:

```
lbl.grid(column=0, row=0)
```

Entonces el código completo se verá de esta manera:

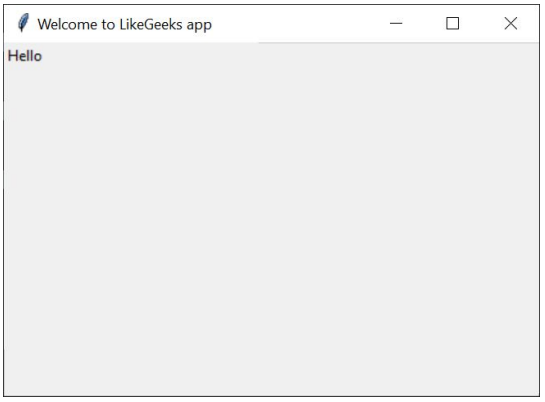
```
from tkinter import *

window = Tk()

window.title("Welcome to LikeGeeks app")

lbl = Label(window, text="Hello")

lbl.grid(column=0, row=0)

window.mainloop()
```

Establecer el tamaño de una ventana

Podemos establecer el tamaño predeterminado de ventana, usando la función `geometry`, de esta manera:

```
window.geometry('350x200')
```

La línea anterior establece el ancho de la ventana en 350 píxeles y la altura en 200 píxeles.



Puedes establecer la fuente de la etiqueta para agrandarla o quizá, colocarla en negrita. También puedes cambiar el estilo de fuente.

Para hacerlo, puedes pasar el parámetro `font` de esta manera:

```
lbl = Label(window, text="Hello", font=("Arial Bold", 50))
```

Intentemos agregar más widgets como botones y ver cómo manejar el evento de `click` de un botón.

Agregar un widget button

Comencemos agregando un botón a la ventana. El botón se crea y se agrega a la ventana de la misma manera que la etiqueta:

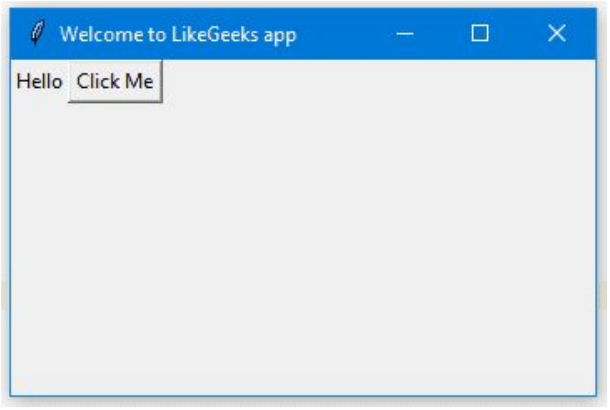
```
btn = Button(window, text="Click Me")

btn.grid(column=1, row=0)
```

Entonces, el código de la ventana se verá así:

```
from tkinter import *
```

```
window = Tk()

window.title("Welcome to LikeGeeks app")

window.geometry('350x200')

lbl = Label(window, text="Hello")

lbl.grid(column=0, row=0)

btn = Button(window, text="Click Me")

btn.grid(column=1, row=0)

window.mainloop()
```

Y el resultado se verá así:



Hay que tener en cuenta que colocamos el botón en la segunda columna de la ventana, que es la 1. Si olvidas eso y colocas el botón en la misma columna (en este caso la 0), se mostrará el botón solamente, ya que el botón estará por encima de la etiqueta.

Manejar el evento click de un botón

Primero, escribiremos la función que necesitamos ejecutar cuando se haga click en el botón:

```
def clicked():

    lbl.configure(text="Button was clicked !!")
```

Luego la cableamos con el botón especificando la función de esta manera:

```
btn = Button(window, text="Click Me", command=clicked)
```

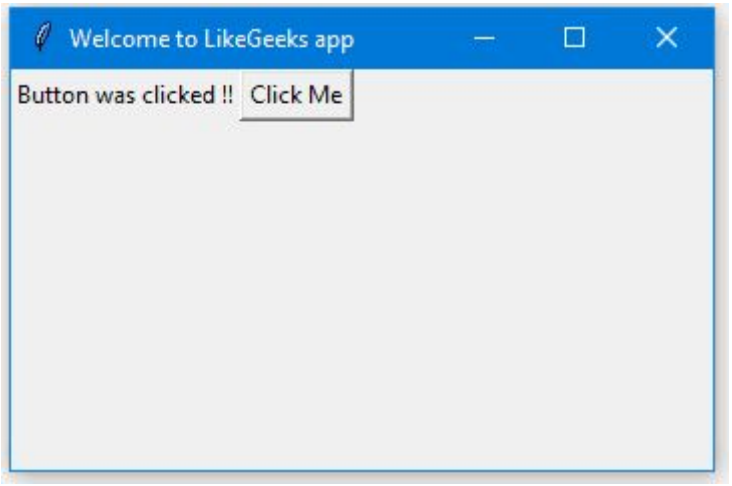
Fíjate que, escribimos `clicked` simplemente y no `clicked()` con paréntesis.

Ahora el código completo se verá de esta manera:

```
from tkinter import *

window = Tk()

window.title("Welcome to LikeGeeks app")

window.geometry('350x200')

lbl = Label(window, text="Hello")

lbl.grid(column=0, row=0)

def clicked():

    lbl.configure(text="Button was clicked !!")

btn = Button(window, text="Click Me", command=clicked)

btn.grid(column=1, row=0)
```

```
window.mainloop()
```

Y cuando hacemos click en el botón, el resultado es el esperado:



Entrada de datos usando la clase Entry (Tkinter textbox)

En los ejemplos anteriores de Python GUI, vimos cómo agregar widgets simples, ahora intentemos obtener una entrada del usuario utilizando la clase Tkinter Entry (cuadro de texto de Tkinter).

Puedes crear un cuadro de texto usando la clase Tkinter Entry de esta manera:

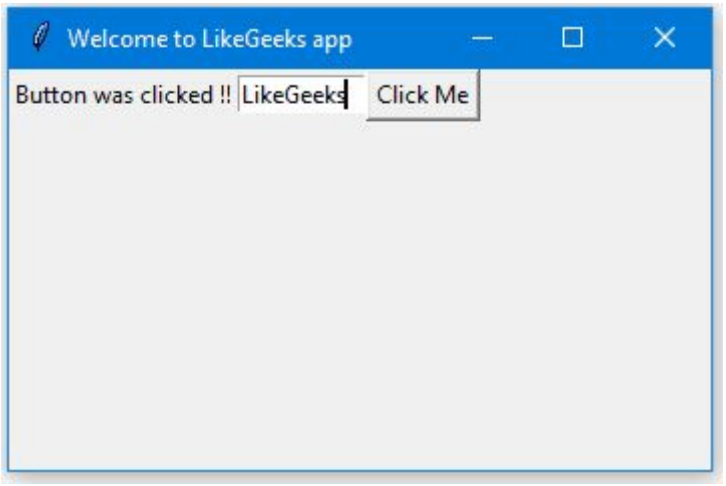
```
txt = Entry(window,width=10)
```

Luego, puedes agregar el widget a la ventana usando la función `grid`, como siempre.

Entonces nuestra ventana será así:

```
from tkinter import *

window = Tk()

window.title("Welcome to LikeGeeks app")

window.geometry('350x200')

lbl = Label(window, text="Hello")

lbl.grid(column=0, row=0)

txt = Entry(window,width=10)

txt.grid(column=1, row=0)

def clicked():

    lbl.configure(text="Button was clicked !!")

btn = Button(window, text="Click Me", command=clicked)

btn.grid(column=2, row=0)

window.mainloop()
```

Y el resultado será el siguiente:



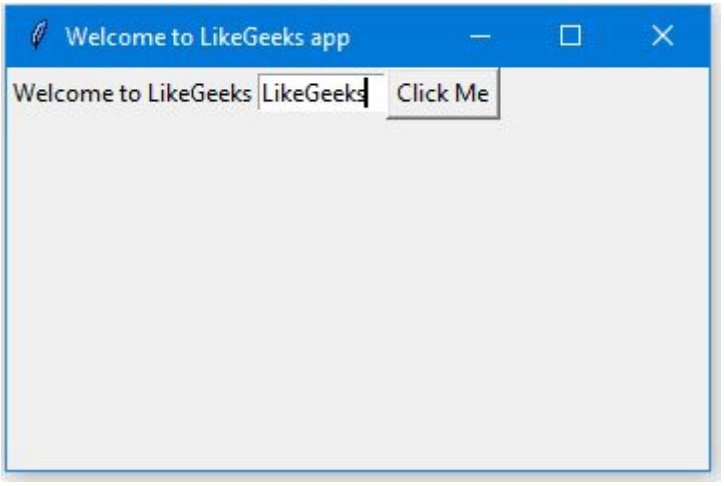
Ahora, si haces click en el botón, se mostrará el mismo mensaje anterior. ¿Qué tal si se muestra el texto ingresado en el widget de entrada?

Primero, podemos obtener texto de entrada usando la función `get`. Entonces podemos escribir este código en nuestra función `clicked`, de esta manera:

```
def clicked():

    res = "Welcome to " + txt.get()

    lbl.configure(text= res)
```

Si haces click en el botón y hay un texto en el widget de entrada, se mostrará “Welcome to ” concatenado con el texto ingresado.

Ejecutando el código el resultado es:



¡Increíble!

Agregar widgets radio button

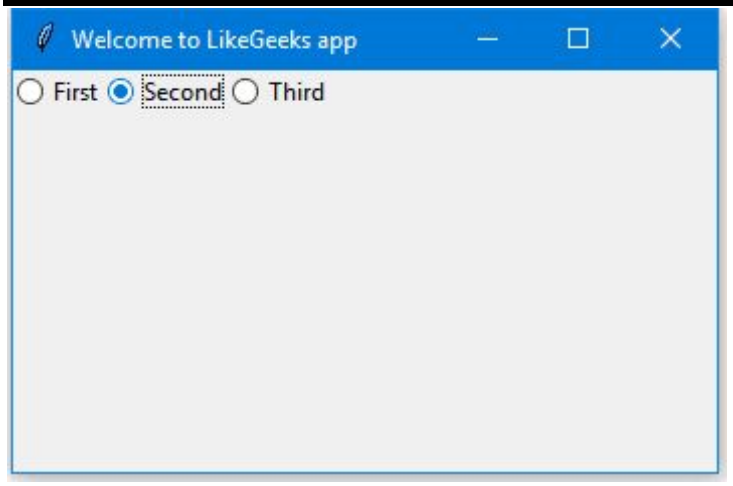
Para agregar radio buttons, simplemente puedes usar la clase `RadioButton`, de esta manera:

```
rad1 = Radiobutton(window,text='First', value=1)
```

Ten en cuenta que, debes establecer el valor para cada botón de opción con un valor diferente; de lo contrario, no funcionarán.

```
from tkinter import *

from tkinter.ttk import *

window = Tk()

window.title("Welcome to LikeGeeks app")

window.geometry('350x200')
```

```
rad1 = Radiobutton(window,text='First', value=1)

rad2 = Radiobutton(window,text='Second', value=2)

rad3 = Radiobutton(window,text='Third', value=3)

rad1.grid(column=0, row=0)

rad2.grid(column=1, row=0)

rad3.grid(column=2, row=0)

window.mainloop()
```



Además, puedes configurar el `command` de cualquiera de estos botones de opción para una función específica, de modo que si el usuario hace click en alguno de ellos, ejecuta el código de la función asignada.

Aquí un ejemplo:

```
rad1 = Radiobutton(window,text='First', value=1,
command=clicked)
```

```
def clicked():


# Do what you need
```

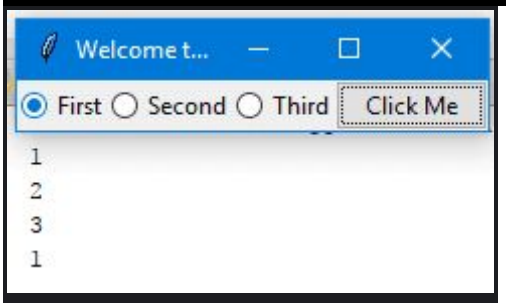
¡Bastante simple!

Obtener el valor del radio button (opción seleccionada)

Para obtener el botón de opción actualmente seleccionado o el valor del botón de opción, puedes pasar un parámetro variable a los botones de opción y más adelante puede obtener su valor.

```
from tkinter import *

from tkinter.ttk import *

window = Tk()

window.title("Welcome to LikeGeeks app")

selected = IntVar()

rad1 = Radiobutton(window,text='First', value=1,
variable=selected)

rad2 = Radiobutton(window,text='Second', value=2,
variable=selected)

rad3 = Radiobutton(window,text='Third', value=3,
variable=selected)

def clicked():
```

```
    print(selected.get())

btn = Button(window, text="Click Me", command=clicked)

rad1.grid(column=0, row=0)

rad2.grid(column=1, row=0)

rad3.grid(column=2, row=0)

btn.grid(column=3, row=0)

window.mainloop()
```



Cada vez que seleccione un botón de opción, el valor de la variable cambiará al valor de la opción seleccionada.

Crear un MessageBox

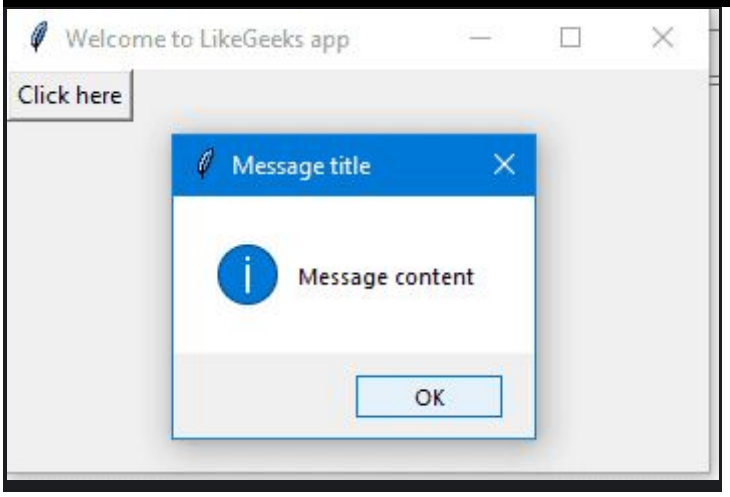
Para mostrar un cuadro de mensaje usando Tkinter, puedes usar la librería `messagebox`, de esta manera:

```
from tkinter import messagebox

messagebox.showinfo('Message title','Message content')
```

¡Bastante simple!

Vamos a mostrar un cuadro de mensaje cuando el usuario haga click en un botón.

```
from tkinter import *

from tkinter import messagebox

window = Tk()

window.title("Welcome to LikeGeeks app")

window.geometry('350x200')

def clicked():

    messagebox.showinfo('Message title', 'Message
content')

btn = Button(window,text='Click here', command=clicked)

btn.grid(column=0,row=0)

window.mainloop()
```



Cuando hagas click en el botón, un cuadro de mensajes informativo aparecerá.

4. Configuración

Para configurar un widget, simplemente llamamos a .config() y pasamos los argumentos que queramos modificar. Algunas opciones son:

- *bg*: modifica el color de fondo. Se puede indicar con el color en inglés (incluyendo modificadores, como “darkgreen”) o su código RGB en hexadecimal (“#aaaaaa” para blanco). **Ojo**: en MacOS no se puede modificar el color de fondo de los botones; aunque indiquemos un nuevo color, se mostrará en blanco. Lo más parecido que podemos hacer es configurar el *highlightbackground*, que pintará el fondo alrededor del botón del color que indiquemos.
- *fg*: cambia el color del texto.
- *cursor*: modifica la forma del cursor. Algunos de los más utilizados son “gumby”, “pencil”, “watch” o “cross”.
- *height*: altura en líneas del componente.
- *width*: anchura en caracteres del componente.
- *font*: nos permite especificar, en una tupla con nombre de la fuente, tamaño y estilo, la fuente a utilizar en el texto del componente. Por ejemplo, Font(“Times New Roman”, 24, “bold underline”).
- *bd*: modificamos la anchura del borde del widget.

- *relief*: cambiamos el estilo del borde del componente. Su valor puede ser “flat”, “sunken”, “raised”, “groove”, “solid” o “ridge”.
- *state*: permite deshabilitar el componente (state=DISABLED); por ejemplo, una *Label* en la que no se puede escribir o un *Button* que no se puede clickar.
- *padding*: espacio en blanco alrededor del widget en cuestión.
- *command*: de cara a que los botones hagan cosas, podemos indicar qué función ejecutar cuando se haga click en el mismo.

5. Gestión de la composición

Es **MUY IMPORTANTE** que, cuando tengamos configurado el componente, utilizamos un gestor de geometría de componentes. Si no, el widget quedará creado pero no se mostrará.

Los tres más conocidos son:

- ***Pack***: cuando añadimos un nuevo componente, se “hace hueco” a continuación de los que ya están incluidos (podemos indicar que se inserte en cualquiera de las 4 direcciones), para finalmente calcular el tamaño que necesita el widget padre para contenerlos a todos.
- ***Place***: este es el más sencillo de entender, pero puede que no el más sencillo de utilizar para todo el mundo. Al insertar un componente, podemos indicar explícitamente la posición (coordenadas X e Y) dentro del widget padre, ya sea en términos absolutos o relativos.

- ***Grid***: la disposición de los elementos es una matriz, de manera que para cada uno debemos indicar la celda (fila y columna) que queremos que ocupe. Podemos además especificar que ocupe más de una fila y/o columna (rowspan/columnspan=3), “pegarlo” a cualquiera de los 4 bordes de la celda en vez de centrarlo (sticky=W para el borde izquierdo, por ejemplo)…

6. Ejecución

Una vez tenemos todos los componentes creados, configurados y añadidos en la estructura, debemos terminar el script con la instrucción *tk.mainloop()* (“tk” = variable Tk). Así, cuando lo ejecutemos, se abrirá la ventana principal de nuestra GUI.

Truco: si guardamos nuestro script con formato .***pyw*** en vez de .py, al ejecutarlo se abrirá nuestra interfaz, sin tener que pasar por terminal o abrir algún IDE para ello.