INTERFAZ GRÁFICA DE USUARIO: TKINTER

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

Python por defecto instala el módulo 'tkinter' para la implementación de interfaces visuales. El editor IDLE que viene con Python está desarrollado utilizando este módulo.

Veremos en una serie de conceptos como implementar interfaces visuales con ventanas, botones, checkbox, label etc. haciendo uso del módulo 'tkinter'.

Lo primero que hay que tener en cuenta es que el módulo 'tkinter' (**tk inter**face) viene instalado por defecto con Python.

Vamos a mostrar una ventana y que en su título aparezca el mensaje 'Hola Mundo'

Los pasos:

1. Importamos el módulo tkinter y le definimos un alias luego de la palabra clave as. Es más fácil luego escribir 'tk' en lugar de 'tkinter':

2. El módulo **'tkinter'** tiene una clase llamada **'Tk'** que representa una ventana. Debemos crear un objeto de dicha clase

3. Llamamos al método **'title**' y le pasamos un string con el mensaje que queremos que aparezca en la barra del título de la ventana

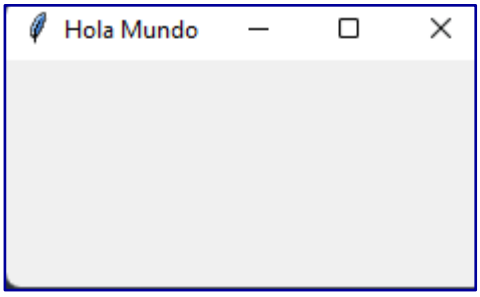
4. Para que se muestre la ventana en el monitor debemos llamar por último al método **'mainloop'** que pertenece a la clase Tk

```
import tkinter as tk

class Aplicacion:
    def __init__(self):
        self.ventana1=tk.Tk()
        self.ventana1.title("Hola Mundo")
        self.ventana1.mainloop()


aplicacion1=Aplicacion()
```

(Ejemplo":tkinter 1.py)



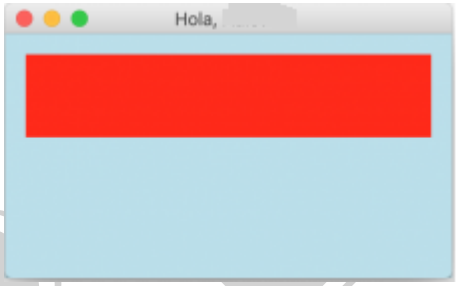
INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

WIDGETS

A la hora de crear 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.

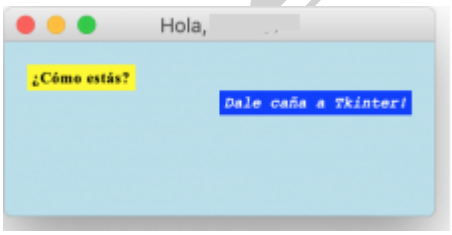
FRAME: El widget Frame es muy importante para el proceso de agrupar y organizar otros widgets de una manera amigable. Funciona como un contenedor, que se encarga de organizar la posición de otros widgets.



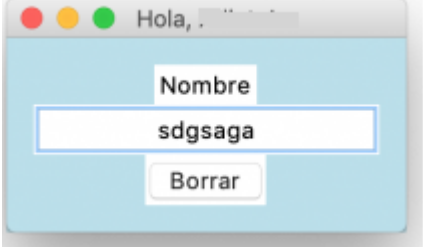
- ***Frame*:** marco que permite agrupar diferentes widgets.

BOOTON Y LABEL

1. Definimos un atributo en la **clase Aplicacion** y almacenamos el valor 1:
2. Creamos la ventana y le fijamos un **título** como hemos visto en el concepto anterior:
3. Creamos un objeto de la **clase Label** y le pasamos como primer parámetro la referencia a la ventana donde debe aparecer la **label** y el parámetro **text** con el valor inicial de la **Label:**
4. Para ubicar los controles visuales en la ventana **Layout**, por ahora en forma muy sencilla mediante la llamada al método **grid** indicaremos en los parámetros **column** y **row** la ubicación del mismo:
5. Para que el texto de la **Label** se muestre de color rojo llamamos al método **configure** y le pasamos en el parámetro **foreground** el **string** "red":
6. La creación de los dos botones es similar a la creación de la label, en el primer parámetro indicamos la referencia de la ventana donde se debe mostrar el botón, en el parámetro text indicamos el texto a mostrar dentro del botón y finalmente en el parámetro **command** pasamos la referencia del método que se ejecutará cuando el operador lo presione:



- ***Label*:** etiqueta estática que permite mostrar texto o imagen.
- ***Button*:** ejecuta una función al ser pulsado.

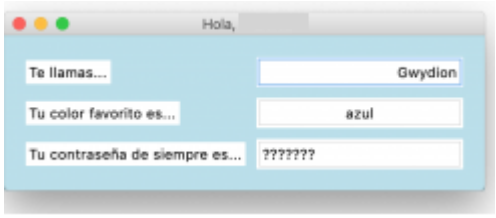


(Ejemplo: TKinter2.py)

INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

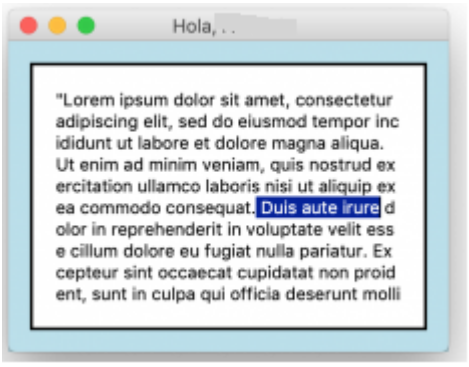
ENTRY

En tkinter el control de entrada de datos por teclado se llama ***Entry***. Con este control aparece el típico recuadro que cuando se le da foco aparece el cursor en forma intermitente esperando que el operador escriba algo por teclado.



- ***Entry***: etiqueta que permite introducir texto corto (típico de formularios).

(Ejemplo: TKinter3.py)



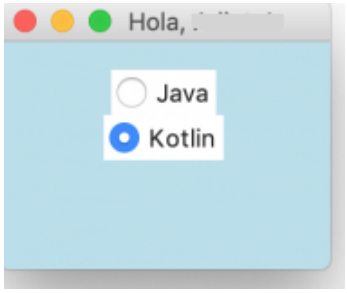
- *Text*: campo que permite introducir texto largo (típico para añadir comentarios).

RADIOBUTTON Y CHECKBUTTON

Normalmente se muestran un conjunto de ***Radiobutton*** y permiten la selección de solo uno de ellos. Se los debe agrupar para que actúen en conjunto, es decir cuando se selecciona uno automáticamente se deben deseleccionar los otros.
Iniciamos el parámetro variable con la referencia de un objeto de la **clase IntVar,** que se encuentra en el módulo tk:

- ***Radiobutton*****:** permite elegir una opción entre varias.

(Ejemplo:TKinter4.py)



- ***Checkbutton*****:** permite elegir varias de las opciones propuestas.

(Ejemplo: TKinter5.py)

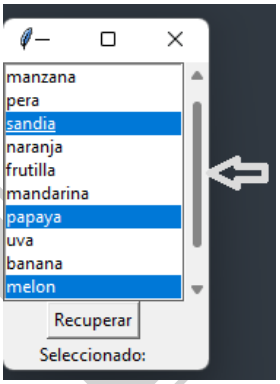
INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

LISTBOX Y BARRA DE SCROLL

Listbox se emplea para mostrar una lista de items. El usuario puede seleccionar uno o más elementos de la lista según como se lo configure al crearlo.

Por defecto no aparece una **barra de scroll** si la cantidad de item supera el tamaño del cuadro del **Listbox**. Para que se muestre una barra de scroll la debemos crear y enlazar con el Listbox.

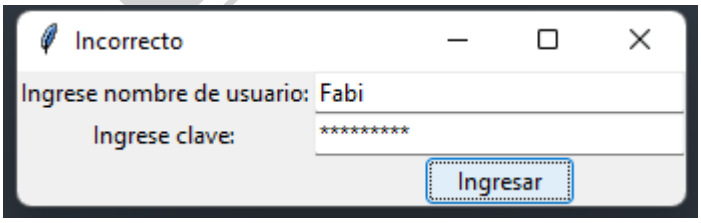
(Ejemplo TKinter6.py)



MÓDULO TTK.

La versión Tk 8.5 sumó una nueva serie de controles visuales (Notebook, Combobox etc.) y modernizó los que hemos visto en los conceptos anteriores. Para hacer uso de esta nueva versión de la biblioteca en Python se implementó un nuevo módulo y se lo agregó al paquete tkinter.
Para hacer uso de este conjunto de Widget (controles visuales) debemos importar el paquete **ttk.**
Todo lo que conocemos hasta ahora de los controles visuales del módulo tkinter funciona prácticamente sin cambios, ***lo que deberemos hacer es crear objetos de la clase Button, Entry etc. recuperándolos ahora del módulo tkinter.ttk***

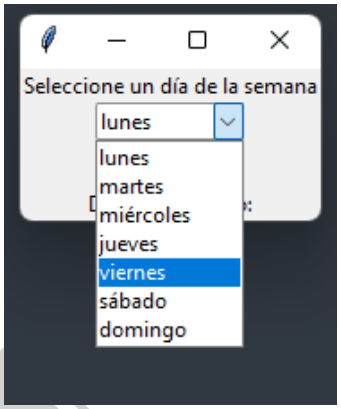
(Ejemplo: TKinter7.py)



INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

COMBOBOX

El control **Combobox** del paquete **ttk** permite seleccionar un string de un conjunto de items que se despliegan.
Podemos indicar cual elemento muestre por defecto mediante el **método 'current'.**
Por defecto el operador puede además de seleccionar un elemento cargar por teclado cualquier cadena.



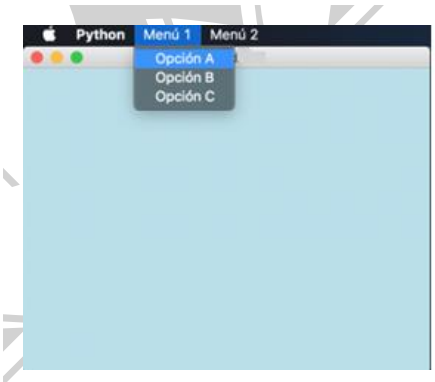
(Ejemplo: TKinter8.py)

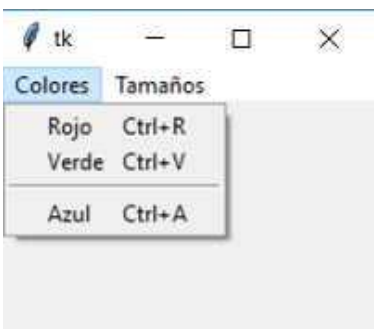
MENÚ

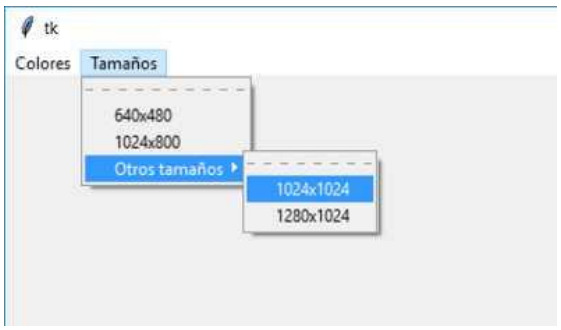
Para implementar los típicos menús de barra horizontales que aparecen en las aplicaciones cuando utilizamos la librería Tk necesitamos crear objetos de la **clase Menu** que ***se encuentra declarada en el paquete tkinter y no en el paquete tkinter.ttk***.
Por lo tanto, Importamos solo el módulo tkinter ya que no utilizamos controles del módulo tkinter.ttk:

Menu: clásico menú superior con opciones (Archivo, Editar…).
(Ejemplo: TKinter9.py)

Otra posibilidad es agregar teclas de acceso rápido a opciones de nuestro menú. Esto se resuelve la parte visual agregando el parámetro '***acelerator***', y por otro lado asignar a cada una de las teclas de acceso rápido la ejecución de un método

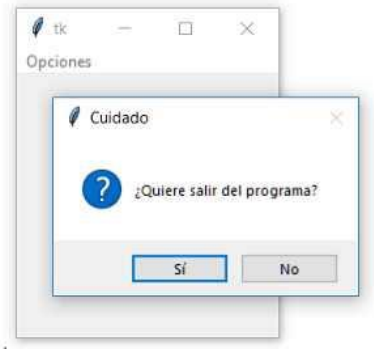




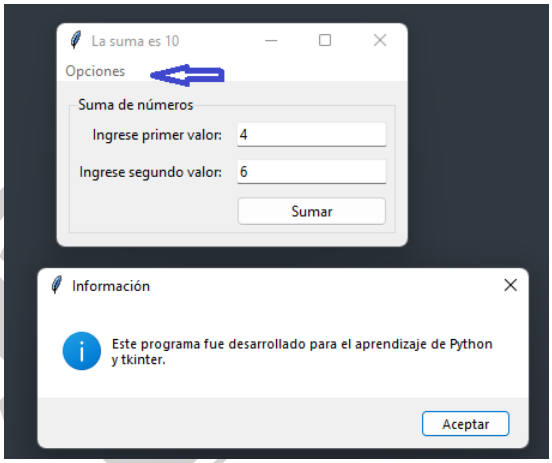


INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

MESSAGEBOX: VENTANAS DE MENSAJES



Es muy común la necesidad de abrir otras ventanas emergentes con el objetivo de informar, advertir de errores etc. La librería tkinter provee un paquete llamado ***messagebox*** con una serie de funciones para la apertura de diálogos de información. Para usar estos diálogos lo primero que debemos hacer es importar el paquete:

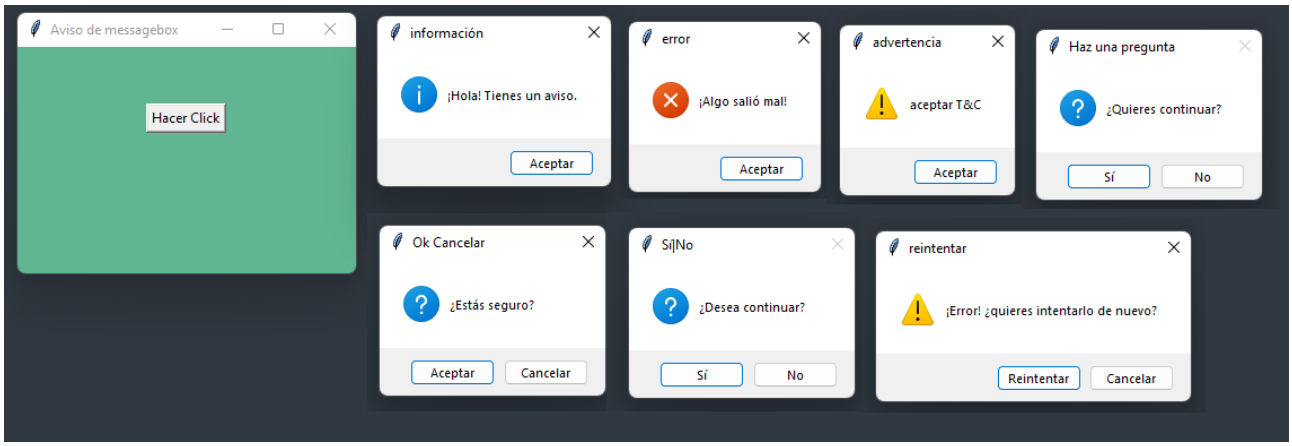


Messagebox proporciona principalmente 6 tipos de avisos de mensajes como ***showinfo(), showerror(), showwarning(), askquestion(), askokcancel(), askyesno(), askretyrcancel().***

showinfo()	se usa cuando se necesita mostrar cualquier confirmación/información. como inicio de sesión exitoso, mensaje
showerror()	utilizado para mostrar un aviso de error con sonido. El momento ideal para su aparición es cuando el usuario comete algún error o se salta el paso necesario.
showwarning()	Muestra un mensaje de advertencia con un signo de exclamación. Advierte al usuario sobre los próximos errores.
askquestion()	Pregunta al usuario por Sí o No. También devuelve 'Sí' o 'No'.
askokcancel()	Pregunta al usuario por Sí o No. También devuelve 'Sí' o 'No'Solicita 'Ok' o 'Cancelar', devuelve 'Verdadero' o 'Falso'
askyesno()	Solicita 'Sí' o 'No'. Devuelve Verdadero para 'Sí' y Falso para 'No'
askyesnocancel()	Solicita 'Sí', 'No' o 'Cancelar'. Sí devuelve Verdadero, No devuelve Falso y Cancelar devuelve Ninguno
askretrycancel()	Solicita las opciones Reintentar y Cancelar. Reintentar devuelve Verdadero y Cancelar devuelve Falso.

INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

(Ejemplo: TKinter11.py)

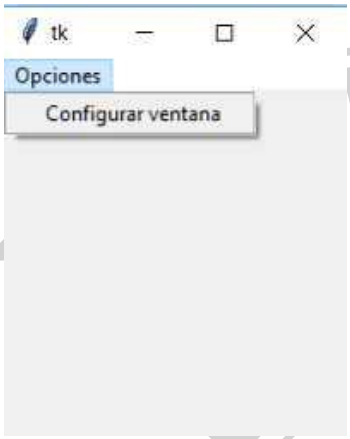


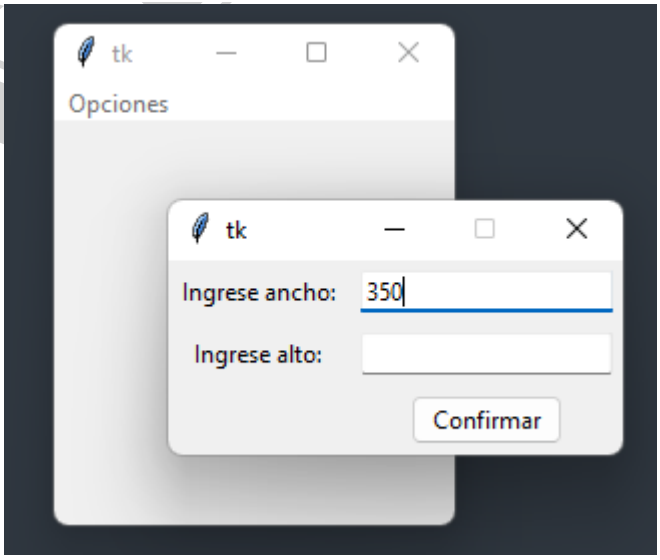
VENTANAS DE DIÁLOGO

Para crear diálogos en tkinter debemos crear un objeto de la clase TopLevel y pasar como parámetro la referencia de la ventana principal.

- ***Dialogs*:** ventana emergente (o *pop-up*).

(Ejemplo: TKinter12.py)





INTERFAZ GRÁFICA DEL USUARIO (GUI): TKINTER

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.

GESTIÓN DE LA COMPOSICIÓN

Es **MUY IMPORTANTE** que, cuando tengamos configurado el componente, utilicemos 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) …

Es preferible utilizar Grid: de esta manera te aseguras la distribución de los componentes, y la hora de añadir nuevos es muy visual poder pensar en ello como matriz, en vez de coordenadas o posiciones relativas a otros widgets.

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.

Bricotruco*:** 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.