2023

Tekinter paso a paso

Este tutorial es un resumen, del curso impartido por Manuel González, de su canal de YouTube Programar en Python es como un juego.

En el código QR podrás acceder a todos su tutoriales.

Pere Manel Verdugo Zamora

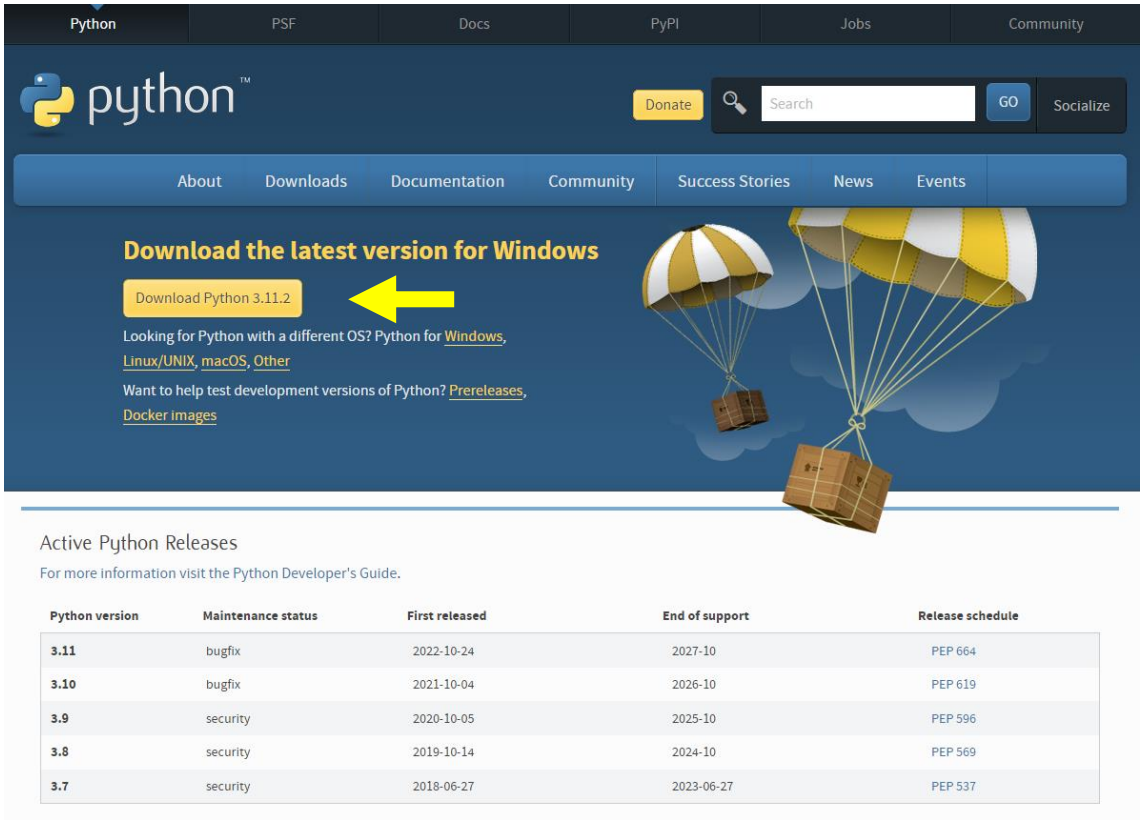
www.peremanelv.com

13-2-2023

1. Introducción

Antes de empezar con este tutorial, vamos a realizar los siguientes pasos:

1.- Actualizar Python desde el siguiente enlace: https://www.python.org/downloads/



En mi caso como yo estoy trabajando con Windows descargaré la opción que me sugiere.

Una vez descargado:

Procederemos a su instalación haciendo doble clic.

Es importante que al inicio de la instalación actives las correspondientes casillas de verificación

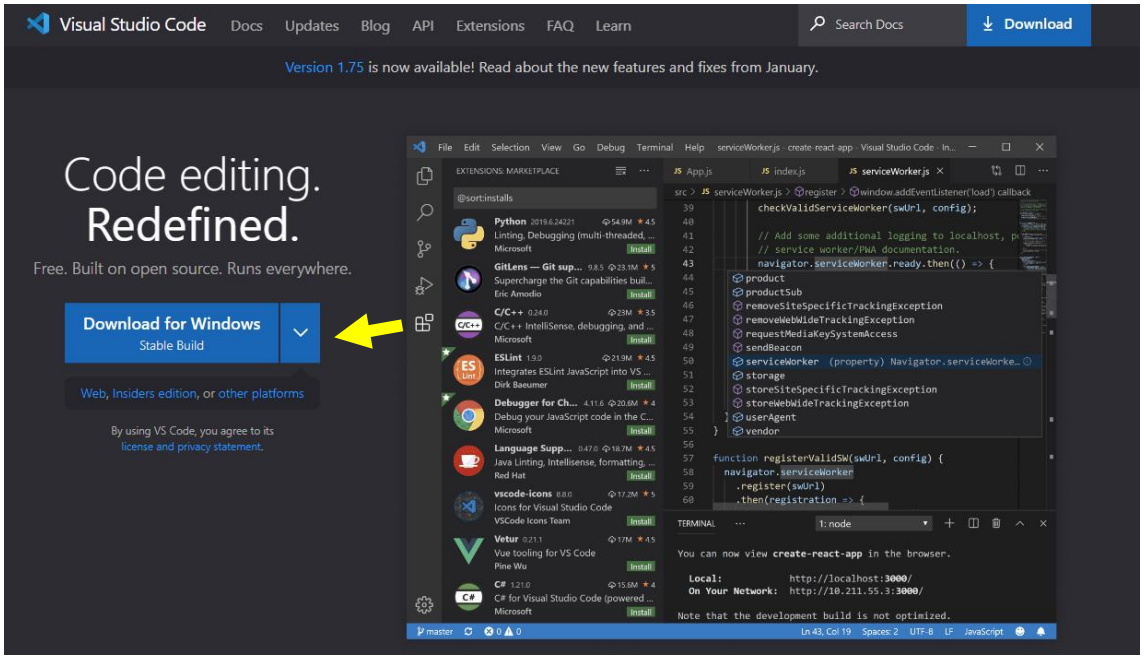
☑ Install launcher for all users (recommended)

☑ Add Python 3.11 to PATH

Esta última opción es necesaria para que puedas ejecutar Python desde cualquier directorio donde estés ubicado.

Para realizar los pasos de este tutorial vamos a utilizar un editor este se llama Visual Studio Code que es totalmente gratis y lo podrás descargar desde este enlace:

https://code.visualstudio.com/



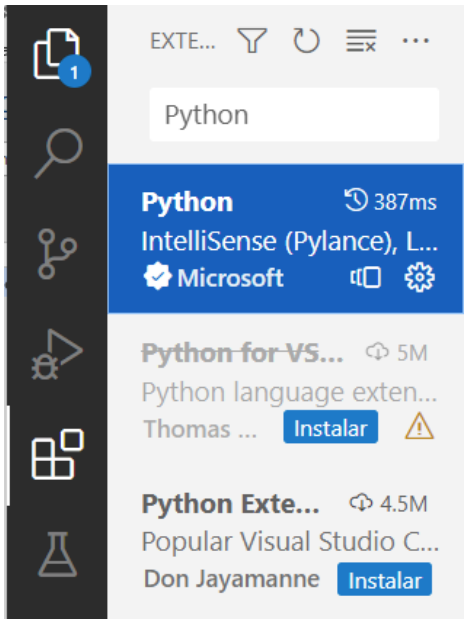
Procederemos a su descarga:

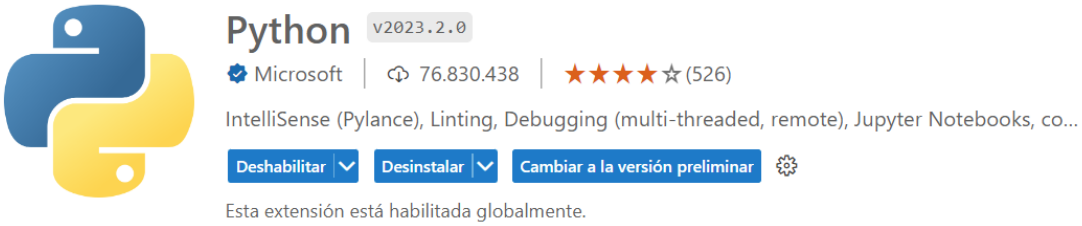
Haciendo doble clic sobre el empezaremos su instalación, te tienes que limitar a seguir los pasos.

Una vez lo tengamos instalado tendremos que instalar una extensión para que nos ayude a la programación en Python.

En la parte izquierda vamos a encontrar el siguiente botón:

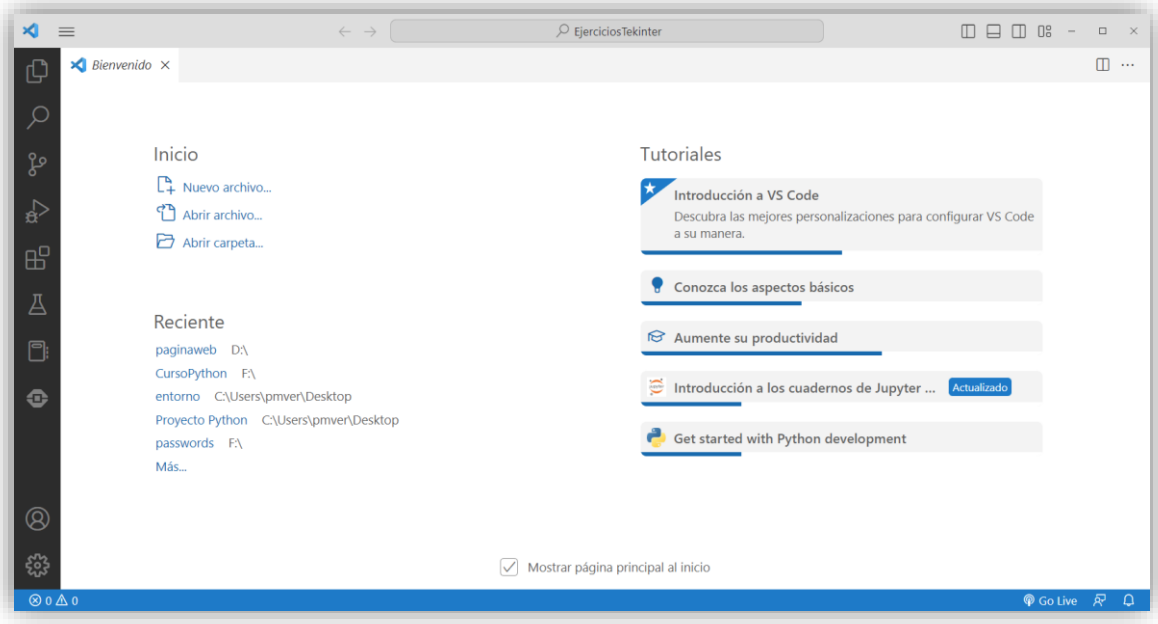
Una vez seleccionado vamos a buscar la correspondiente extensión escribiendo Python.



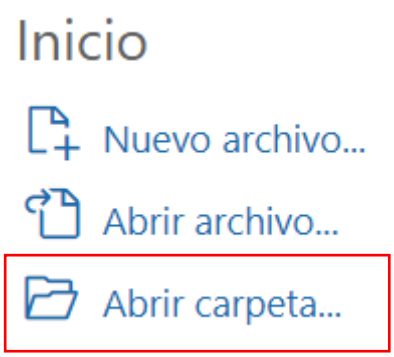


A ti te tiene que aparecer el botón de instalar, a mí no me sale porque ya lo tengo instalado.

Te aconsejo que crees una carpeta en tu ordenador para que en esta se te guarde el proyecto.

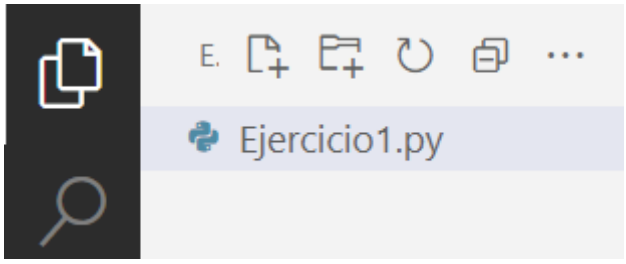


Cuando ejecutes la primera vez selecciona abrir carpeta y selecciona la carpeta que con anterioridad creaste.

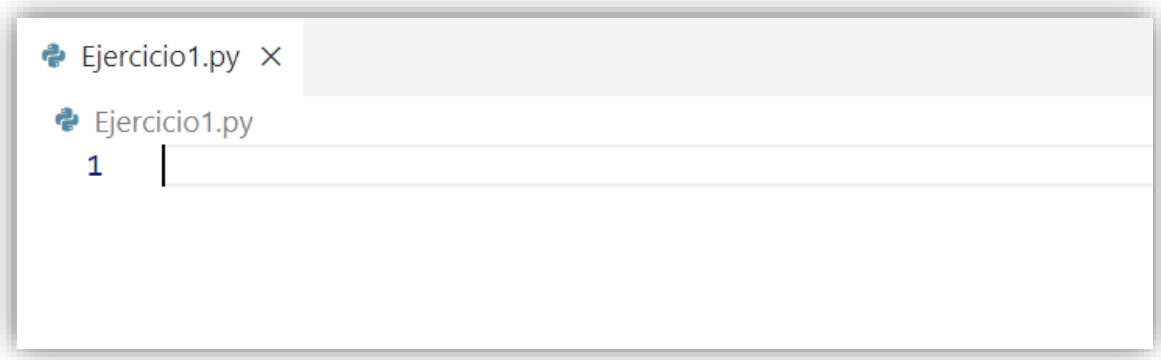


En la parte izquierda si seleccionas el correspondiente botón:

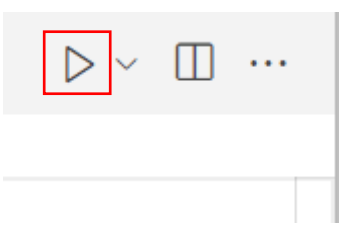
Hay una opción que nos permitirá crear ficheros de Python, cuando los crees recuerda que tienen que tener una extensión py, por ejemplo ejercicio1.py.



Ya podemos empezar a programa:



Una vez hayas finalizado el programa en la parte derecha encontrarás el correspondiente botón para su ejecución.



2.- Primera ventana en tkinter

Lo primero que tenemos que hacer es importar la librería tkinter.

```
1    import tkinter as tk
```

Importamos la librería tkinter con el alias tk de este modo en lugar de escribir tkinter utilizaremos la palabra tk.

También podemos importar de la siguiente forma:

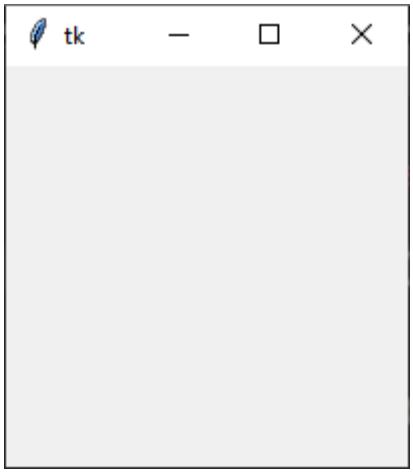
```
from tkinter import *
```

De este forma cargaríamos todos los módulos, todas las clases que contiene esta librería pero en esta ocasión vamos a utilizar el primer ejemplo.

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4
5
6    ventana.mainloop()
```

En la línea 3 creamos la ventana y en la línea 6 le decimos con mainloop() que vaya realizando continuamente una iteración o bucle entre ellas y así se irá repiento todo lo que se encuentra entre ambas instrucciones.

Vamos a ejecutar para ver el resultad0.

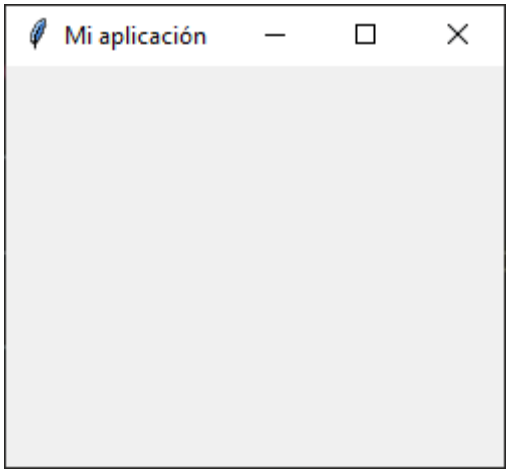


Ya hemos creado nuestra primera ventana.

Vamos a dar unas características a la ventana:

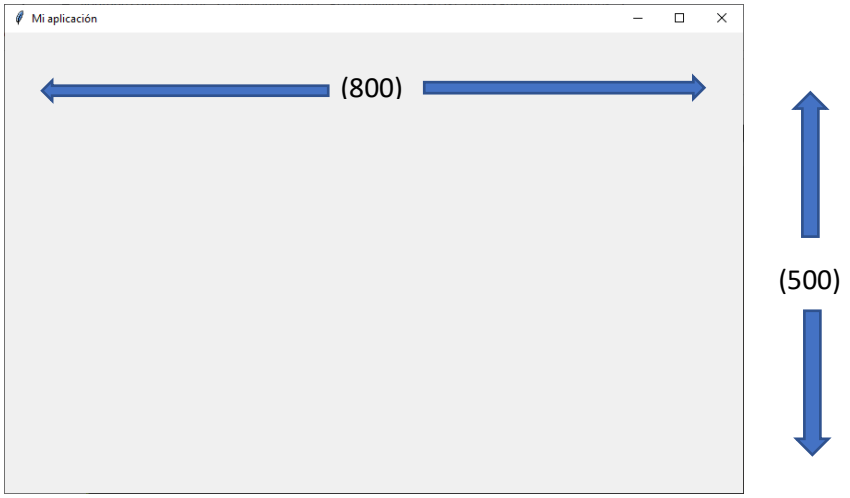
```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5
6    ventana.mainloop()
```

Le hemos puesto un título a la ventana:



Podemos darle una dimensiones a la ventana:

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500")
6
7    ventana.mainloop()
```



Si ejecutamos varias veces podremos observar que la ventana aparece de una forma aleatoria por toda nuestra pantalla, podemos controlar este comportamiento y que siempre aparezca en el mismo lugar.

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500+300+200")
6
7    ventana.mainloop()
```

Después de las dimensiones de la ventana con un + le damos la coordenada x de la pantalla y con otro + la coordenada y.

Ahora podrás observar que siempre se abre en el mismo lugar.

Ahora queremos controlar la redimensión de la ventana:

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500+300+200")
6    ventana.resizable(width=False, height=False)
7
8    ventana.mainloop()
```

Hemos a width (ancho) sea igual a False y de height (alto) igual a false, de este modo si intentamos modificar el ancho y el alto de la ventana no nos va a dejar, si cambiamos alguno por True, a continuación podremos modificarlo, y no se especifica por defecto los dos valores están en True.

Vamos a comentar la línea 6 para poder dimensionar la ventana pero con unos márgenes en concreto.

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500+300+200")
6    # ventana.resizable(width=False, height=False)
```

```
7    ventana.minsize(width=300, height=200)
8
9    ventana.mainloop()
```

Le estamos diciendo que al modificar la ventana a un tamaño menor, no nos dejará que sea menor de ancho 300 y menor de alto 200.

También podemos poner un margen a las dimensiones máximas.

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500+300+200")
6    # ventana.resizable(width=False, height=False)
7    ventana.minsize(width=300, height=200)
8    ventana.maxsize(width=1000, height=700)
9
10   ventana.mainloop()
```

Cuando modifiquemos el tamaño de la ventana a más grande como tome será 1000 de ancho y 700 de alto.

Vamos a configurar ciertos aspectos de la ventana como puede ser el color.

```
1    import tkinter as tk
2
3    ventana = tk.Tk()
4    ventana.title("Mi aplicación")
5    ventana.geometry("800x500+300+200")
6    # ventana.resizable(width=False, height=False)
7    ventana.minsize(width=300, height=200)
8    ventana.maxsize(width=1000, height=700)
9    ventana.configure(background="blue")
10
11   ventana.mainloop()
```

Tipos de colores que podemos utilizar:

black	bisque	lightgreen	slategrey
k	darkorange	forestgreen	lightsteelblue
dimgray	burlywood	limegreen	cornflowerblue
dimgrey	antiquewhite	darkgreen	royalblue
grey	tan	green	ghostwhite
gray	navajowhite	g	lavender
darkgrey	blanchedalmond	lime	midnightblue
darkgray	papayawhip	seagreen	navy
silver	moccasin	mediumseagreen	darkblue
lightgray	orange	springgreen	mediumblue
lightgrey	wheat	mintcream	blue
gainsboro	oldlace	mediumspringgreen	b
whitesmoke	floralwhite	mediumaquamarine	slateblue
white	darkgoldenrod	aquamarine	darkslateblue
w	goldenrod	turquoise	mediumslateblue
snow	cornsilk	lightseagreen	mediumpurple
rosybrown	gold	mediumturquoise	blueviolet
lightcoral	lemonchiffon	azure	indigo
indianred	khaki	lightcyan	darkorchid
brown	palegoldenrod	paleturquoise	darkviolet
firebrick	darkkhaki	darkslategray	mediumorchid
maroon	ivory	darkslategrey	thistle
darkred	beige	teal	plum
red	lightyellow	darkcyan	violet
r	lightgoldenrodyellow	c	purple
mistyrose	olive	cyan	darkmagenta
salmon	y	aqua	m
tomato	yellow	darkturquoise	fuchsia
darksalmon	olivedrab	cadetblue	magenta
coral	yellowgreen	powderblue	orchid
orangered	darkolivegreen	lightblue	mediumvioletred
lightsalmon	greenyellow	deepskyblue	deeppink
sienna	chartreuse	skyblue	hotpink
seashell	lawngreen	lightskyblue	lavenderblush
chocolate	sage	steelblue	palevioletred
saddlebrown	lightsage	aliceblue	crimson
sandybrown	darksage	dodgerblue	pink
peachpuff	honeydew	lightslategrey	lightpink
peru	darkseagreen	lightslategray	
linen	palegreen	slategray	

Por último indicar que hay algunos métodos que podemos simplificar:

configure ➔ config.

background ➔ bg.

3.- Rótulos o etiquetas con el widget Label

Ya hemos visto la creación de la ventana.

El segundo paso será implementar los widgets, que tipos hay, a continuación a estos widgets habrá de darles funcionalidad para que lleve a cabo las tareas que se les quiere asignar, a continuación colocar los widgets (Geometry managers) dentro de la ventana.

Widgets básicos de tkinter:

Rotulo ➔ tkinter,Label()

Botón ➔ tkinter.Button()

Entrada ➔ tkinter.Entry()

Texto ➔ tkinter.Text()

Cuadro ➔ tkinter.Frame()

Casilla ➔ tkinter.Chechbutton()

Opción ➔ tkinter.Radiobitton()

Para este capítulo vamos a partir del siguiente código:

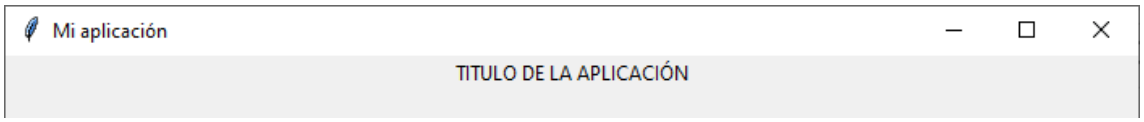
```
1   import tkinter as tk
2
3   ##### VENTANA DEL PROGRAMA #####
4
5   ventana = tk.Tk()
6   ventana.title("Mi aplicación")
7   ventana.geometry("700x500+500+100")
8
9
10  ##### BUCLE PRINCIPAL PROGRAMA #####
11  ventana.mainloop()
```

Geometry manager de tkinter

- widget.pack()
- widget.grid()
- widget.place()

```
1   import tkinter as tk
2
3   ##### VENTANA DEL PROGRAMA #####
4
5   ventana = tk.Tk()
6   ventana.title("Mi aplicación")
7   ventana.geometry("700x500+500+100")
8
9   ##### CREACIÓN DE WIDGETS #####
10
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN")
13  rotulo.pack()
14
15  ##### BUCLE PRINCIPAL PROGRAMA #####
16  ventana.mainloop()
```

Creamos un objeto llamado rotulo de la clase tkinter de tipo Label, que irá en la ventana, con un texto "TÍTULO DE LA APLICACIÓN", que finalmente con pack() lo agregaremos a nuestra ventana.



Vamos a cambiar el tamaño y tipo de letra:

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font="consolas 20 bold")
14  rotulo.pack()
```

En la línea 13 en el caso que queramos modificar el tipo de fuente dinámicamente la escribiremos de la siguiente forma:

```
font=("consolas", 20, "bold")
```

El resultado será el mismo.

Vamos a cambiar el fondo del rótulo:

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font=("consolas", 20, "bold"),
14      background="blue")
15  rotulo.pack()
```

También podemos cambiar el color de la fuente:

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font=("consolas", 20, "bold"),
14      background="blue",
15      foreground="red")
16  rotulo.pack()
```

Ahora cambiar background=lightblue.

Ahora queremos dejar unos márgenes tanto en izquierdo y derecho como arriba y abajo a esto se le denomina padding.

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font=("consolas", 20, "bold"),
14      background="lightblue",
15      foreground="red",
16      padx=20, pady=20)
17  rotulo.pack()
```

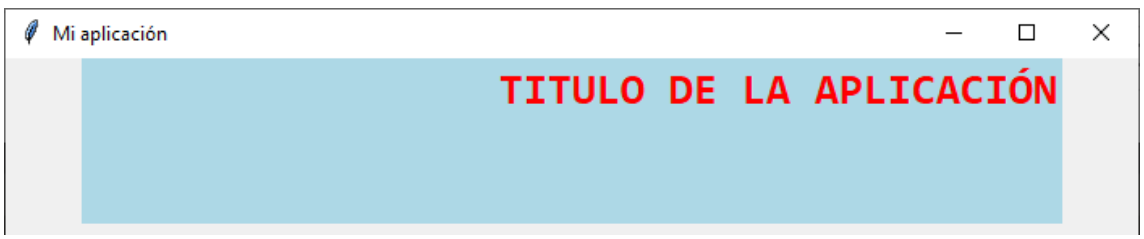
Vamos a comentar la línea 16 para utilizar otra forma, pero en lugar de pixeles, será espacios.

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font=("consolas", 20, "bold"),
14      background="lightblue",
15      foreground="red",
16      # padx=20, pady=20
17      width=40, height=3)
18  rotulo.pack()
```

Los espacios serán dependiendo del tamaño de la fuente.

Otra característica es en qué lado queremos anclar la etiqueta.

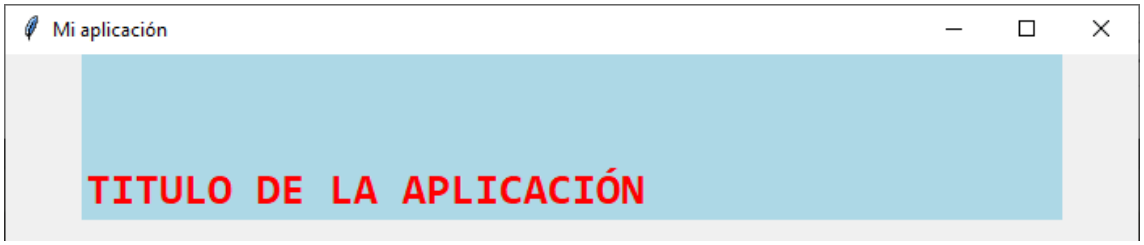
```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN",
13      font=("consolas", 20, "bold"),
14      background="lightblue",
15      foreground="red",
16      # padx=20, pady=20
17      width=40, height=3,
18      anchor="ne")
19  rotulo.pack()
```



Hemos puesto la coordenado Noroeste.

Si lo cambiamos por sw.

```
anchor="sw"
```



Esto se notará en el espacio de dicha etiqueta, si esta tiene menos espacio el cambio se notará menos.

También es correcto de la siguiente forma:

```
anchor=tk.CENTER)
```

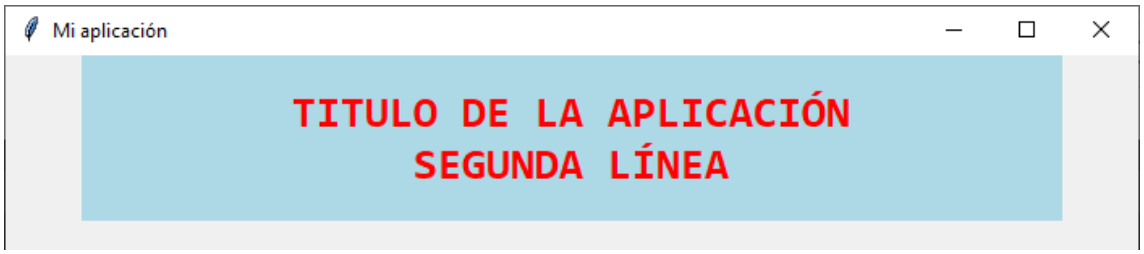
También podemos justificar el texto:

```
11    rotulo = tk.Label(ventana,
12        text="TITULO DE LA APLICACIÓN\nSEGUNDA LÍNEA",
13        font=("consolas", 20, "bold"),
14        background="lightblue",
15        foreground="red",
16        # padx=20, pady=20
17        width=40, height=3,
18        anchor=tk.CENTER,
19        justify="right")
20    rotulo.pack()
```

En la línea 12 con \n agregamos una segunda línea, para poder ver el justificado a la derecha de la línea 19.

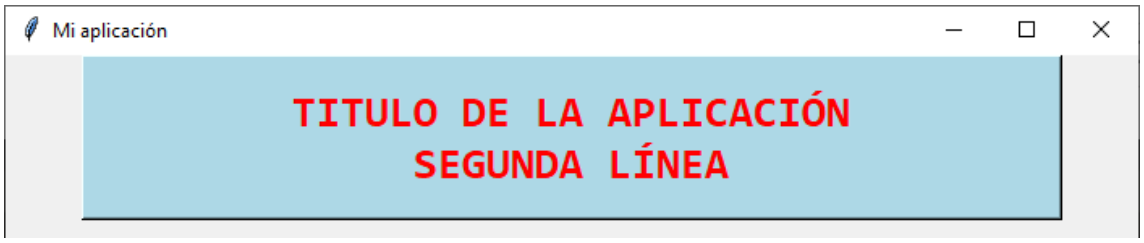
Vamos a cambiar por:

```
justify=tk.CENTER)
```



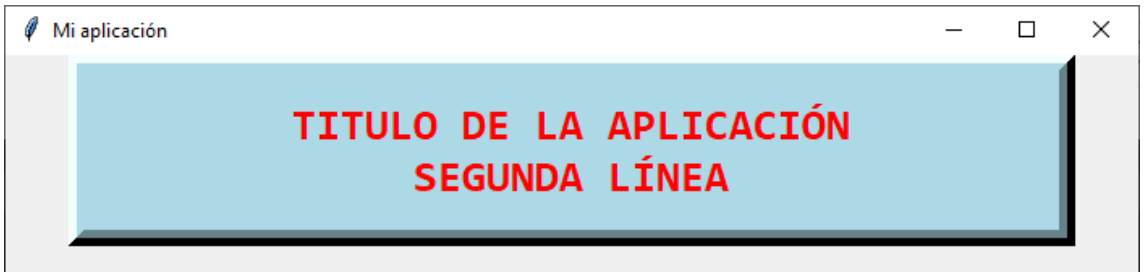
Podemos dar un arco al rótulo:

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN\nSEGUNDA LÍNEA",
13      font=("consolas", 20, "bold"),
14      background="lightblue",
15      foreground="red",
16      # padx=20, pady=20
17      width=40, height=3,
18      anchor=tk.CENTER,
19      justify=tk.CENTER,
20      relief=tk.RAISED)
21  rotulo.pack()
```



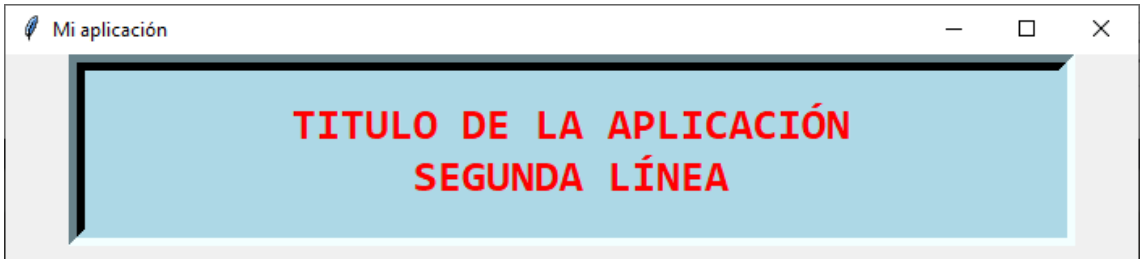
Vamos a agregar un borde.

```
11  rotulo = tk.Label(ventana,
12      text="TITULO DE LA APLICACIÓN\nSEGUNDA LÍNEA",
13      font=("consolas", 20, "bold"),
14      background="lightblue",
15      foreground="red",
16      # padx=20, pady=20
17      width=40, height=3,
18      anchor=tk.CENTER,
19      justify=tk.CENTER,
20      relief=tk.RAISED,
21      border=10)
22  rotulo.pack()
```

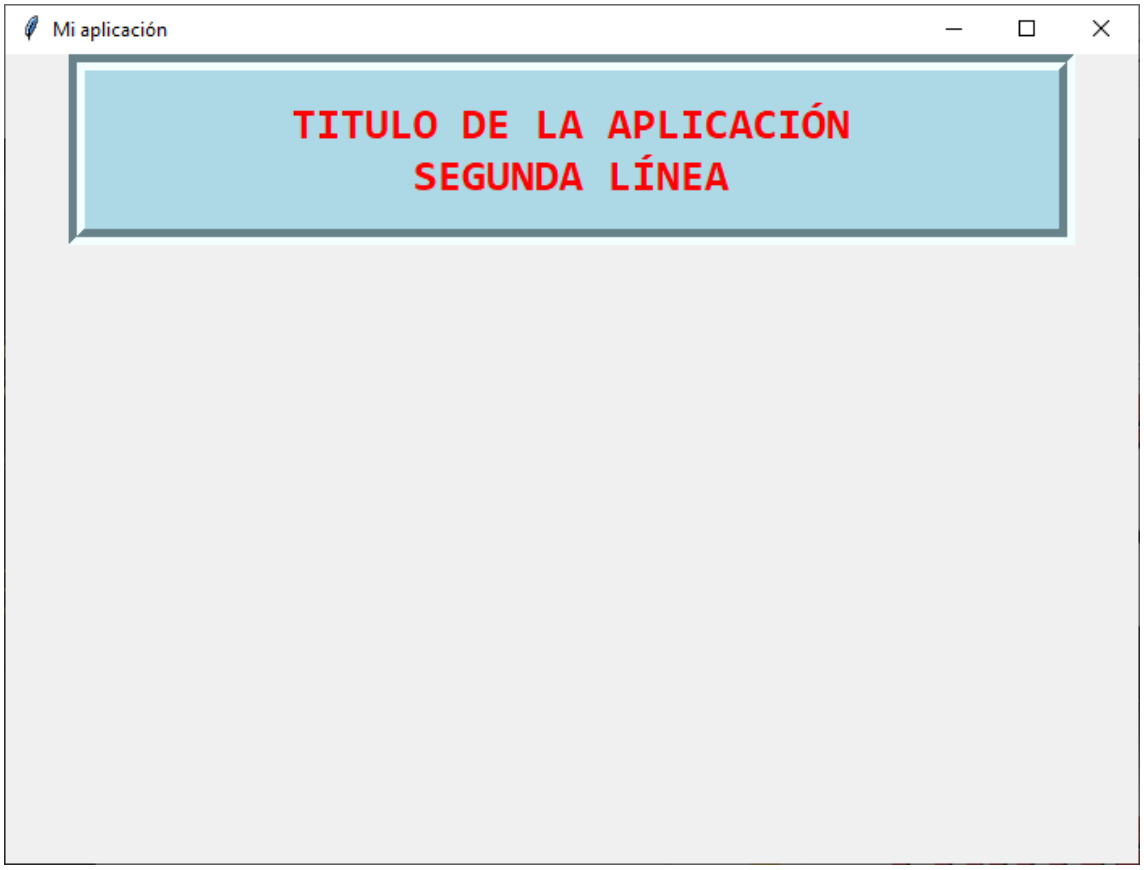


Vamos a cambiar:

```
relief=tk.SUNKEN,
```




```
relief=tk.GROOVE,
```



4.- Botones en tkinter con el widget Button

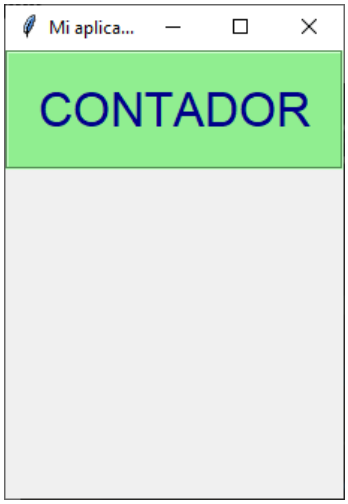
Vamos a realizar un contador tal como se muestra en la siguiente figura:


```
1   import tkinter as tk
2
3   ##### VENTANA DEL PROGRAMA #####
4
5   ventana = tk.Tk()
6   ventana.title("Mi aplicación")
7
8   ##### LABEL TITULO #####
9
10  titulo = tk.Label(ventana,
11      text="CONTADOR",
12      font=("arial", 24),
13      bg="lightgreen", fg="darkblue",
14      relief=tk.GROOVE,
15      padx=20, pady=20
16      )
17  titulo.pack()
18
19  ventana.mainloop()
```

Al quitar las dimensiones de la ventana podemos observar como la ventana se ajusta a la etiqueta.

Vamos a poner dimensiones mínimas.

```
3    ##### VENTANA DEL PROGRAMA #####
4
5    ventana = tk.Tk()
6    ventana.title("Mi aplicación")
7    ventana.minsize(width=200, height=300)
```



Después del título:

```
20   ##### LABEL NUMERO #####
21
22   numero = tk.Label(ventana,
23       text="0",
24       font=("arial", 48),
25       bg="darkgreen", fg="lightblue",
26       padx=20, pady=20
27       )
28   numero.pack()
```

Ya tenemos el numero que hará las funciones de contador.

Vamos a crear el widget del botón después del LABEL NUMERO.

```
30    ##### BUTTON CONTAR #####
31
32    contar = tk.Button(ventana,
33          text="CONTAR",
34          font=("arial", 18),
35          bg="lightgrey", fg="black",
36          padx=20, pady=20,
37          relief="groove", bd=10)
38    contar.pack()
```

5.- Dando funcionalidad a los botones de tkinter con la opción command

Seguimos con el proyecto de capítulo anterior que estábamos realizando un contador.

Si presionamos el botón este no hace nada ya que no le hemos introducido la funcionalidad.

Antes de seguir con el proyecto vamos a realizar una prueba.

```
30    ##### BUTTON CONTAR #####
31
32    contar = tk.Button(ventana,
33          text="CONTAR",
34          font=("arial", 18),
35          bg="lightgrey", fg="black",
36          padx=20, pady=20,
37          relief="groove", bd=10,
38          command=sys.exit)
39    contar.pack()
```

Hemos agregado un funcionalidad de salir del programa, podrás observar que sys parece con un subrayado, esto significa que nos está dando un error.

Para solucionarlo hemos de agregar una librería para que funciones sys.

```
1    import tkinter as tk
2    import sys
```

Debajo de la librería tkinter agregamos la librería sys, podrás observar que el error que había en sys.exit ya no está.

Ahora cuando ejecutemos la aplicación y presionemos el botón este se cerrará.

Este no es el objetivo final, pero te puede ayudar a comprender mejor la función command, esta ejecuta una instrucción o bien una función.

Ahora vamos a crear la función que luego será llamada por el botón cuando hagamos clic.

```
10    ##### FUNCIONES #####
11    contador = 0
12
13    def cuenta_numeros():
14        global contador
15        contador +=1  # CONTADOR SE INCREMENTA EN 1
```

```
16      numero["text"] = contador
```

Línea 11 asignamos a la variable contador el valor de 0.

Línea 13 definimos una función llamada cuenta_numeros()

Línea 14 definimos la variable contador como global para que entre llamada y llamada de la función esta no pierda su valor.

Línea 15 a la variable contador le incrementamos 1, esto puede ser igual a:

contador = contador + 1.

Línea 16 al objeto numero al atributo text le asignamos el valor de contador.

```
38  ##### BUTTON CONTAR #####
39
40  contar = tk.Button(ventana,
41        text="CONTAR",
42        font=("arial", 18),
43        bg="lightgrey", fg="black",
44        padx=20, pady=20,
45        relief="groove", bd=10,
46        command=cuenta_numeros)
47  contar.pack()
```

Desde la propiedad command del botón llamamos a la función cuenta_numeros.

Vamos a ejecutar y presionar el botón 5 veces.

Ya funciona nuestro contador.

Recuerda que desde el atributo command cuando llamamos a la función no tenemos que poner los paréntesis, porque al crearse el botón se ejecuta automáticamente la función y cuando hagamos clic al botón este no se ejecutará.

Vamos a ver otras posibilidades para ejecutar esta tarea.

```
 9    ##### FUNCIONES #####
10
11    def cuenta_numeros():
12        contador = int(numero["text"])
13        contador +=1
14        numero["text"] = contador
```

En la línea 11 definimos la función cuenta_numeros().

En la línea 12 le decimos que contador es igual al valor entero de lo que contiene el objeto número en el atributo “text”.

En la línea 13 a la variable contador se le incrementa 1.

En la línea 14 al objeto numero al atributo text se le asigna el nuevo valor de contador.

Si ejecutas podrás observar que funciona igual que en el ejemplo anterior.

Ahora vamos a cambiar el atributo del objeto numero.

```
 9    ##### FUNCIONES #####
10
11    def cuenta_numeros():
12        contador = int(numero["text"])
13        contador +=1
14        numero.config(text=contador)
```

Con la configuración del objeto número accedemos al atribulo text para asignarle el nuevo valor de contador.

```
27    ##### LABEL NUMERO #####
28
29    numero = tk.Label(ventana,
30        text=0,
31        font=("arial", 48),
32        bg="darkgreen", fg="lightblue",
33        padx=20, pady=20
```

```
34      )
35  numero.pack()
```

En la línea 30 hemos eliminado las comillas para que text tenga un valor entero.

```
 9  ##### FUNCIONES #####
10
11  def cuenta_numeros():
12      contador = numero["text"]
13      contador +=1
14      numero.config(text=contador)
```

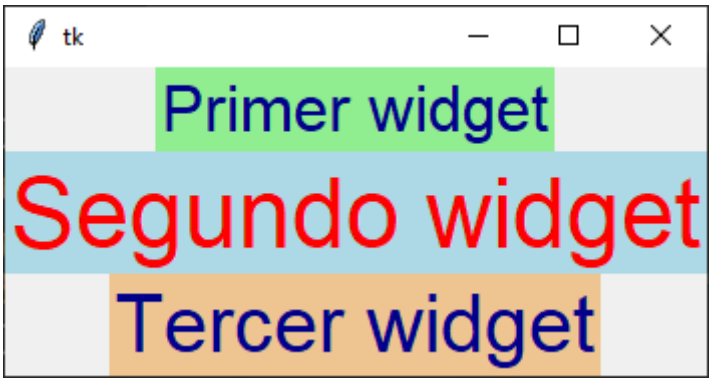
En la función en la línea 12 ya no tenemos la necesidad de reconvertirlo a un número entero , pues podemos eliminar el int(), si ejecutamos este funcionará igualmente.

6.- Método pack com0 geometry manager

.pack(), .grid() y .place()

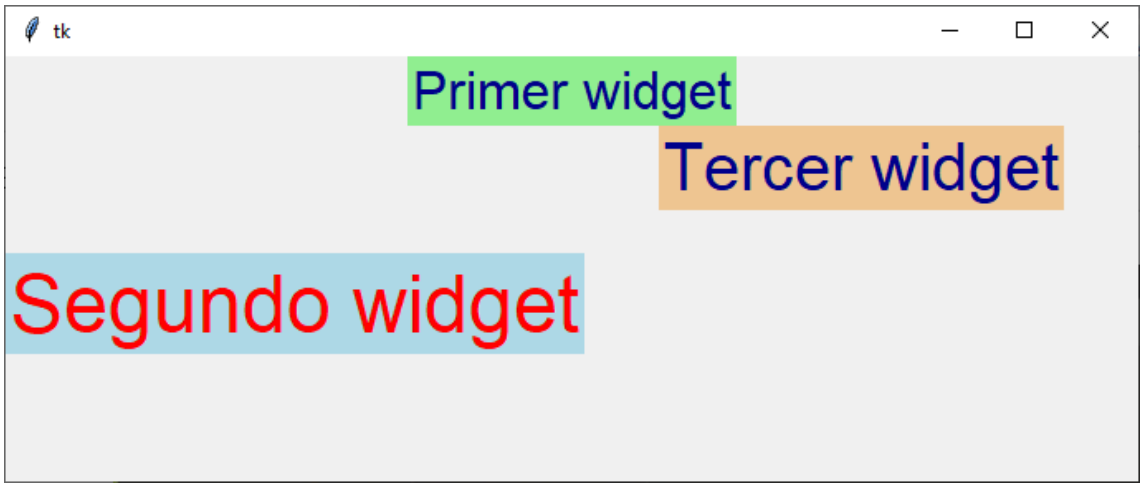
.pack() es uno de los métodos que hemos utilizado con los ejemplos anteriores, este método coloca los widgets uno debajo del otro, en el centro de la pantalla.

```
1   import tkinter as tk
2   ventana = tk.Tk()
3
4   ##### CREACIÓN WIDGETS #####
5
6   rotulo1 = tk.Label(ventana,
7       text="Primer widget",
8       font=("arial", 24),
9       bg="lightgreen", fg="darkblue")
10  rotulo1.pack()
11
12  rotulo2 = tk.Label(ventana,
13      text="Segundo widget",
14      font=("arial", 36),
15      bg="lightblue", fg="red")
16  rotulo2.pack()
17
18  rotulo3 = tk.Label(ventana,
19      text="Tercer widget",
20      font=("arial", 30),
21      bg="burlywood2", fg="darkblue")
22  rotulo3.pack()
23
24  ventana.mainloop()
```



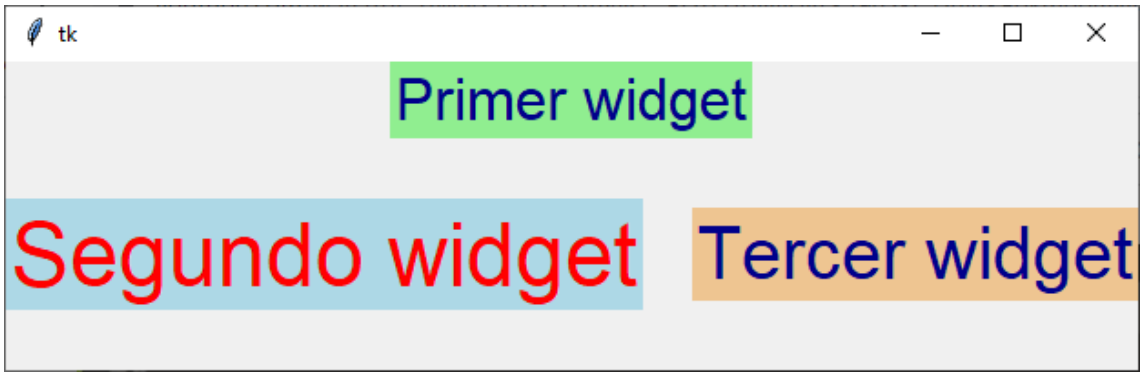
```
12    rotulo2 = tk.Label(ventana,
13        text="Segundo widget",
14        font=("arial", 36),
15        bg="lightblue", fg="red")
16    rotulo2.pack(side=tk.LEFT)
```

Agregamos un parámetro a .pack()




```
18    rotulo3 = tk.Label(ventana,
19        text="Tercer widget",
20        font=("arial", 30),
21        bg="burlywood2", fg="darkblue")
22    rotulo3.pack(side=tk.RIGHT)
```

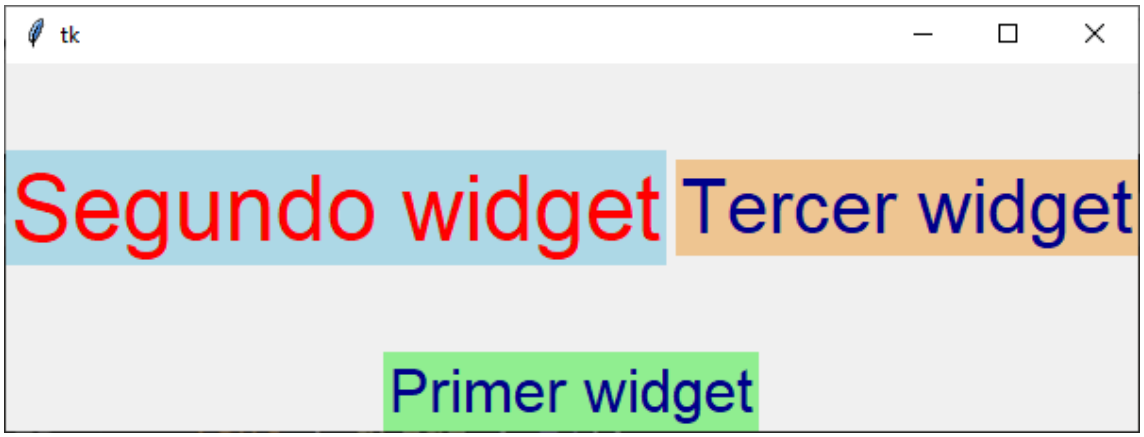
Del rótulo3 cambiamos los parámetros del .pack()




```
6     rotulo1 = tk.Label(ventana,
7         text="Primer widget",
8         font=("arial", 24),
9         bg="lightgreen", fg="darkblue")
```

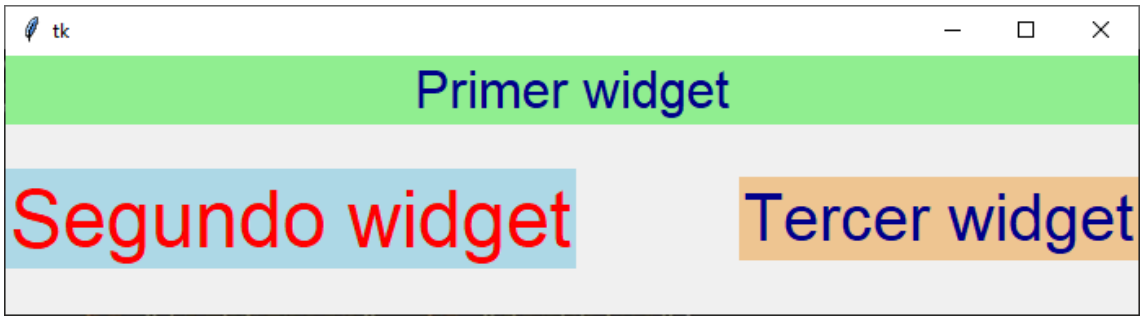
```
10    rotulo1.pack(side=tk.BOTTOM)
```

Modificamos el rotulo1 en el parámetro .pack()



Ahora vamos a ver lo que hace el fill.

```
 6  rotulo1 = tk.Label(ventana,
 7      text="Primer widget",
 8      font=("arial", 24),
 9      bg="lightgreen", fg="darkblue")
10  rotulo1.pack(fill=tk.X)
```



Ocupa todo el ancho.

Vamos a modificar el rótulo2:

```
12  rotulo2 = tk.Label(ventana,
13      text="Segundo widget",
14      font=("arial", 36),
15      bg="lightblue", fg="red")
16  rotulo2.pack(side=tk.LEFT, fill=tk.Y)
```

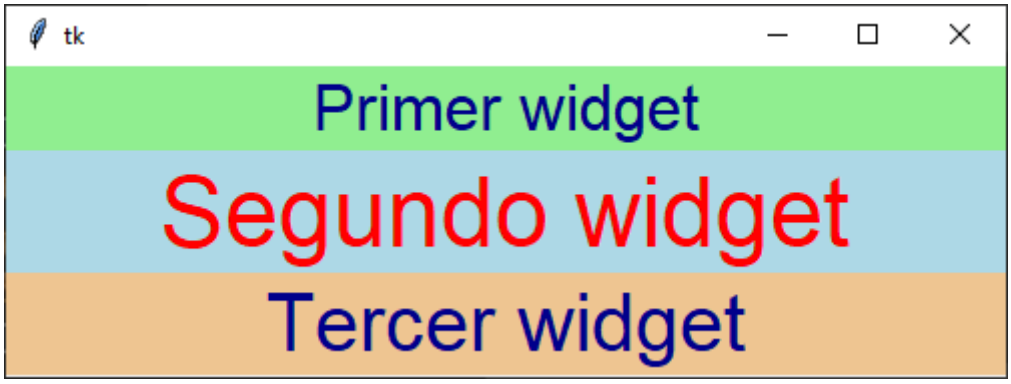
Vamos a modificar el rotulo3.

```
18    rotulo3 = tk.Label(ventana,
19        text="Tercer widget",
20        font=("arial", 30),
21        bg="burlywood2", fg="darkblue")
22    rotulo3.pack(side=tk.BOTTOM, fill=tk.BOTH)
```


```
 6    rotulo1 = tk.Label(ventana,
 7        text="Primer widget",
 8        font=("arial", 24),
 9        bg="lightgreen", fg="darkblue")
10    rotulo1.pack(fill=tk.BOTH)
11
12    rotulo2 = tk.Label(ventana,
13        text="Segundo widget",
```

```
14        font=("arial", 36),
15        bg="lightblue", fg="red")
16    rotulo2.pack(fill=tk.BOTH)
17
18    rotulo3 = tk.Label(ventana,
19        text="Tercer widget",
20        font=("arial", 30),
21        bg="burlywood2", fg="darkblue")
22    rotulo3.pack(fill=tk.BOTH)
```

Modificamos las siguientes líneas:



BOTH ➔ Significa ambos, será ambos lados.

Vamos a modificar el rotulo1.

```
6     rotulo1 = tk.Label(ventana,
7         text="Primer widget",
8         font=("arial", 24),
9         bg="lightgreen", fg="darkblue")
10    rotulo1.pack(fill=tk.BOTH, expand=True)
```

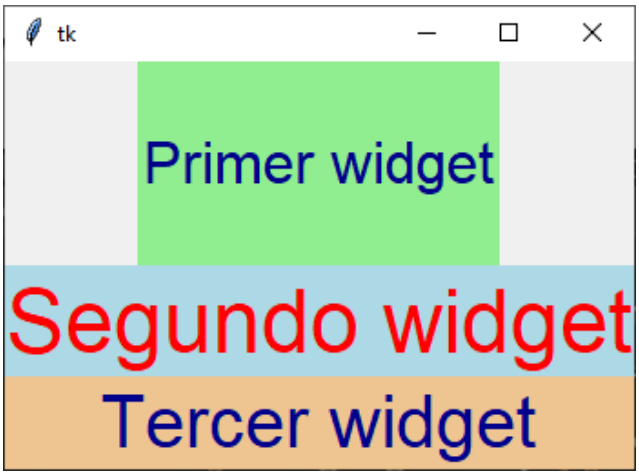
Cuando modificas el tamaño de la ventana el rotulo1 se expande.

```
6  rotulo1 = tk.Label(ventana,
7      text="Primer widget",
8      font=("arial", 24),
9      bg="lightgreen", fg="darkblue")
10 rotulo1.pack(fill=tk.X, expand=True)
```

Si en rotulo1 cambiamos BOTH por X.


```
6  rotulo1 = tk.Label(ventana,
7      text="Primer widget",
8      font=("arial", 24),
9      bg="lightgreen", fg="darkblue")
10 rotulo1.pack(fill=tk.Y, expand=True)
```

Si lo cambiamos la X por la Y.



```
 6   rotulo1 = tk.Label(ventana,
 7       text="Primer widget",
 8       font=("arial", 24),
 9       bg="lightgreen", fg="darkblue")
10   rotulo1.pack(fill=tk.BOTH, expand=True)
11
12   rotulo2 = tk.Label(ventana,
13       text="Segundo widget",
14       font=("arial", 36),
15       bg="lightblue", fg="red")
16   rotulo2.pack(fill=tk.BOTH, expand=True)
17
18   rotulo3 = tk.Label(ventana,
19       text="Tercer widget",
20       font=("arial", 30),
21       bg="burlywood2", fg="darkblue")
22   rotulo3.pack(fill=tk.BOTH, expand=True)
```

Si ponemos los tres expand=True

Esto permite que al redimensionar la ventana se redimensionan los 3 widgets.

Vamos a agregar padx y pady.

```
6  rotulo1 = tk.Label(ventana,
7      text="Primer widget",
8      font=("arial", 24),
9      bg="lightgreen", fg="darkblue")
10 rotulo1.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)
11
12 rotulo2 = tk.Label(ventana,
13     text="Segundo widget",
14     font=("arial", 36),
15     bg="lightblue", fg="red")
16 rotulo2.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)
17
18 rotulo3 = tk.Label(ventana,
19     text="Tercer widget",
20     font=("arial", 30),
21     bg="burlywood2", fg="darkblue")
22 rotulo3.pack(fill=tk.BOTH, expand=True, padx=20, pady=20)
```


```
6  rotulo1 = tk.Label(ventana,
7      text="Primer widget",
8      font=("arial", 24),
9      bg="lightgreen", fg="darkblue")
10 rotulo1.pack(fill=tk.BOTH, expand=True, padx=20, pady=20, ipadx=20, ipady=20)
11
12 rotulo2 = tk.Label(ventana,
13     text="Segundo widget",
14     font=("arial", 36),
15     bg="lightblue", fg="red")
16 rotulo2.pack(fill=tk.BOTH, expand=True, padx=20, pady=20, ipadx=20, ipady=20)
17
```

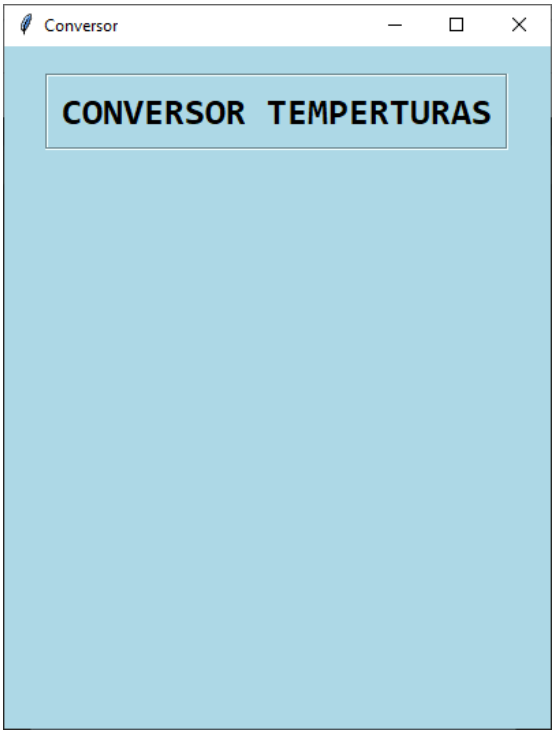
```
18  rotulo3 = tk.Label(ventana,
19      text="Tercer widget",
20      font=("arial", 30),
21      bg="burlywood2", fg="darkblue")
22  rotulo3.pack(fill=tk.BOTH, expand=True, padx=20, pady=20, ipadx=20, ipady=20)
```

Deja dentro del widget un espacio por arriba y por debajo de 20 pixeles.

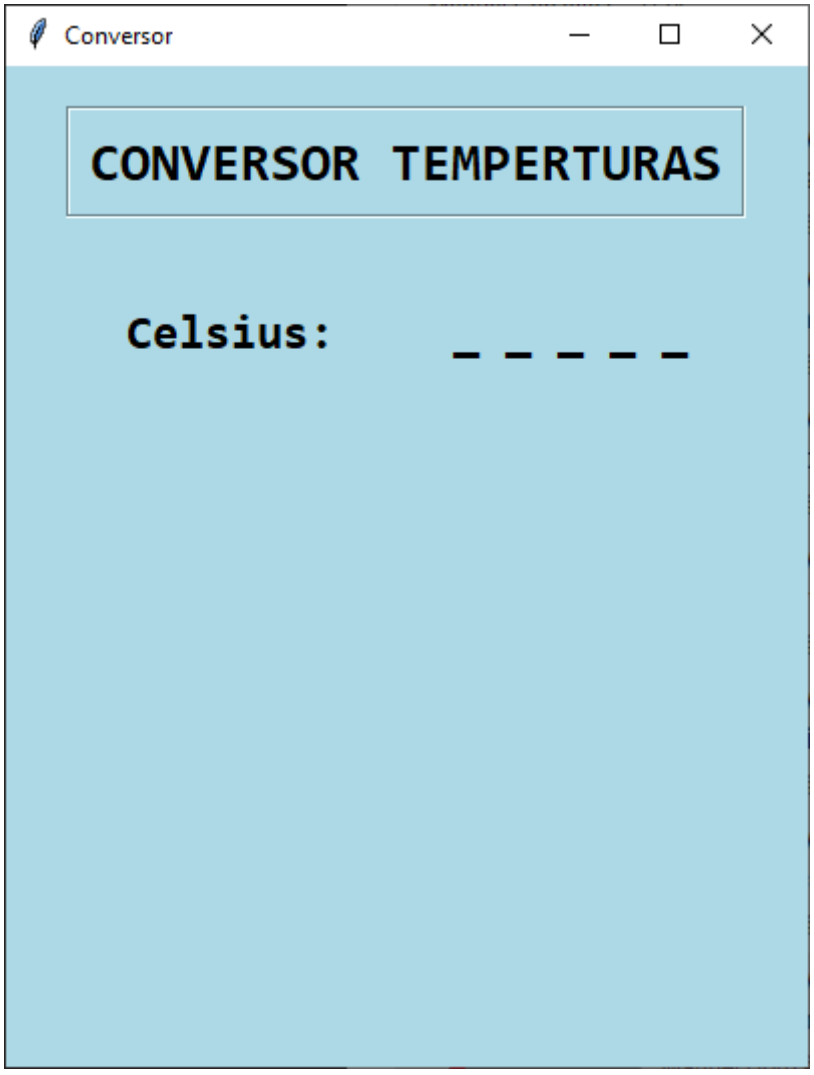
7.- Widget Frame en tkinter par conversor de temperaturas

En este capítulo vamos a realizar un conversor de temperaturas.

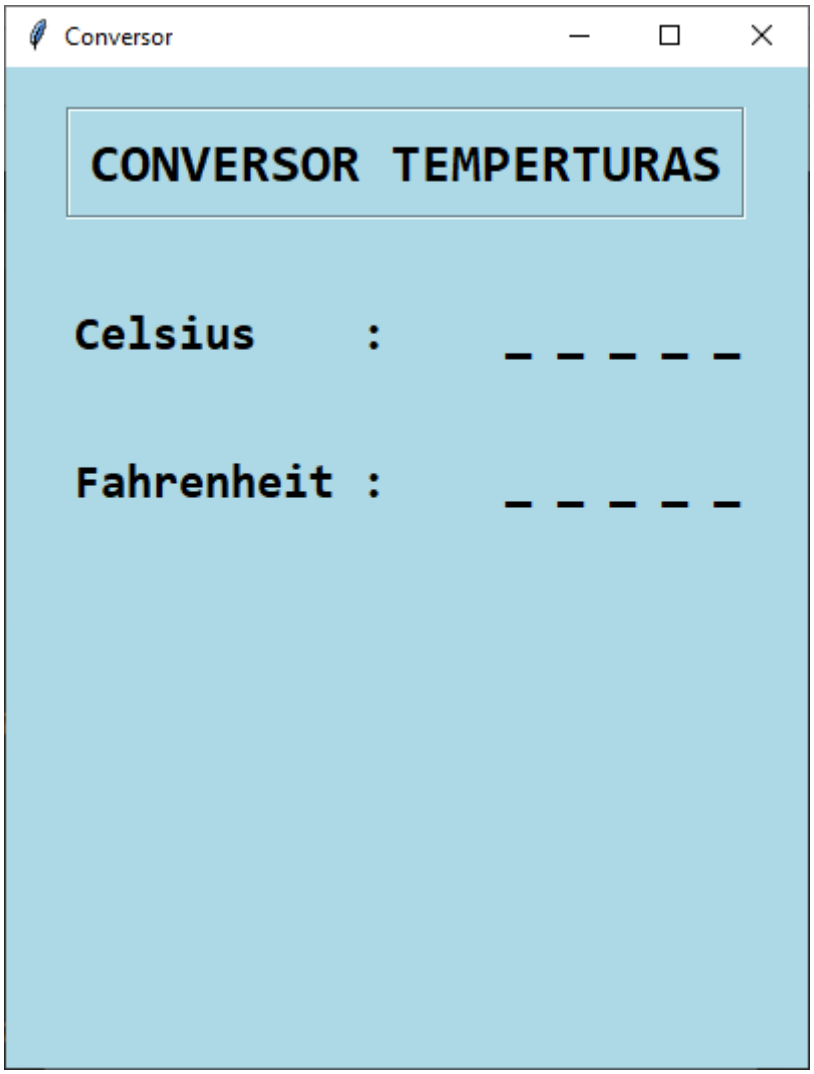
```
1   import tkinter as tk
2
3   ventana = tk.Tk()
4   ventana.title("Conversor")
5   ventana.geometry("400x500+700+100")
6   ventana.minsize(width=400, height=500)
7   ventana.configure(bg="lightblue")
8
9   ##### LABEL TITULO #####
10  rotulo_titulo = tk.Label(ventana,
11      text="CONVERSOR TEMPERTURAS",
12      bg="lightblue", fg="black",
13      font= "consolas 20 bold",
14      relief=tk.GROOVE, bd=2,
15      padx=10, pady=10)
16  rotulo_titulo.pack(padx=20, pady=20)
17
18  ventana.mainloop()
```



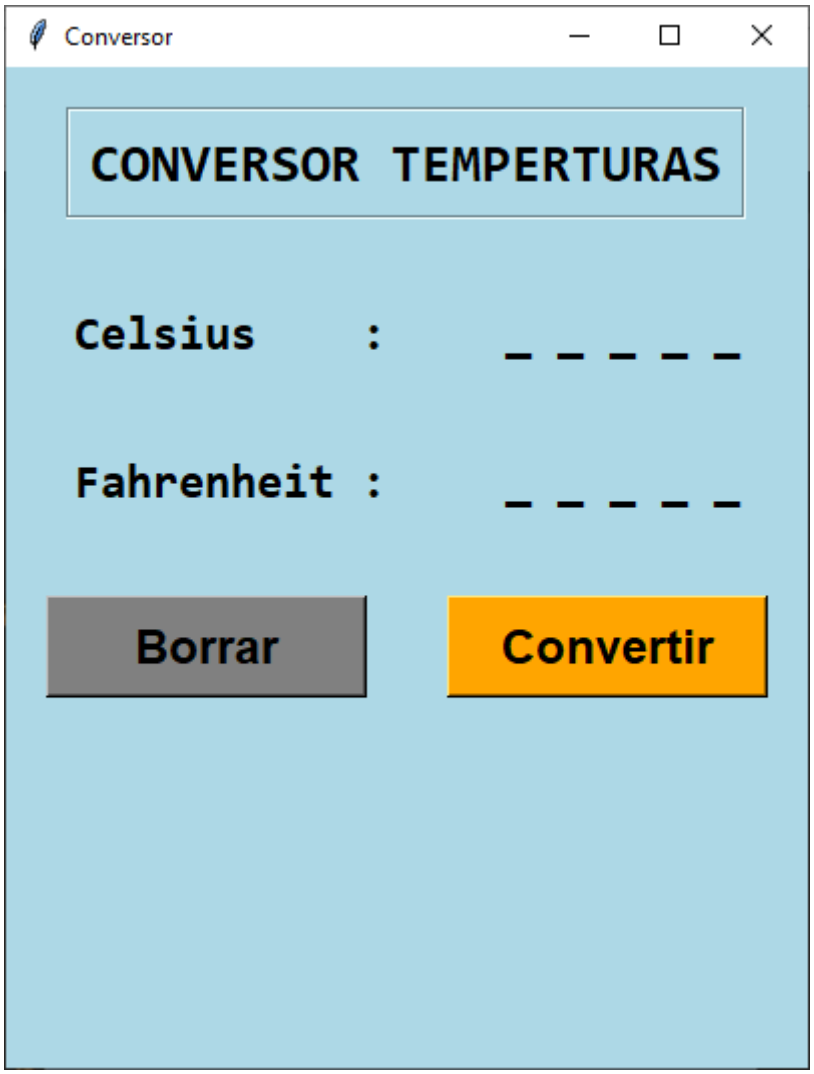
```
18  ##### FRAME PRIMERO #####
19  cuadro1 = tk.Frame(ventana,
20      bg="lightblue")
21  rotulo_celsius = tk.Label(cuadro1,
22      text="Celsius    : ",
23      bg="lightblue",
24      font="consolas 18 bold")
25  rotulo_celsius.pack(side=tk.LEFT, padx=20, pady=20)
26  entrada_celsius = tk.Label(cuadro1,
27      text="_ _ _ _ _",
28      bg="lightblue",
29      font="consolas 18 bold")
30  entrada_celsius.pack(side=tk.LEFT, padx=20, pady=20)
31  cuadro1.pack()
```



```
32  ##### FRAME SEGUNDO #####
33  cuadro2 = tk.Frame(ventana,
34      bg="lightblue")
35  rotulo_fahrenheit = tk.Label(cuadro2,
36      text="Fahrenheit : ",
37      bg="lightblue",
38      font="consolas 18 bold")
39  rotulo_fahrenheit.pack(side=tk.LEFT, padx=20, pady=20)
40  salida_fahrenheit = tk.Label(cuadro2,
41      text="_ _ _ _ _",
42      bg="lightblue",
43      font="consolas 18 bold")
44  salida_fahrenheit.pack(side=tk.LEFT, padx=20, pady=20)
45  cuadro2.pack()
```



```
48  ##### FRAME TERCERO #####
49  cuadro3 = tk.Frame(ventana,
50      bg="lightblue")
51  boton_borrar = tk.Button(cuadro3,
52      text="Borrar",
53      bg="grey",
54      font="consolas, 18 bold",
55      width=10)
56  boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
57  boton_convertir = tk.Button(cuadro3,
58      text="Convertir",
59      bg="orange",
60      font="consolas, 18 bold",
61      width=10)
62  boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
63  cuadro3.pack()
```



8.- Caja de texto con el widget Entry para la entrada de datos.

En este capitulo vamos a ver un widget de entrada de texto de una sola línea.

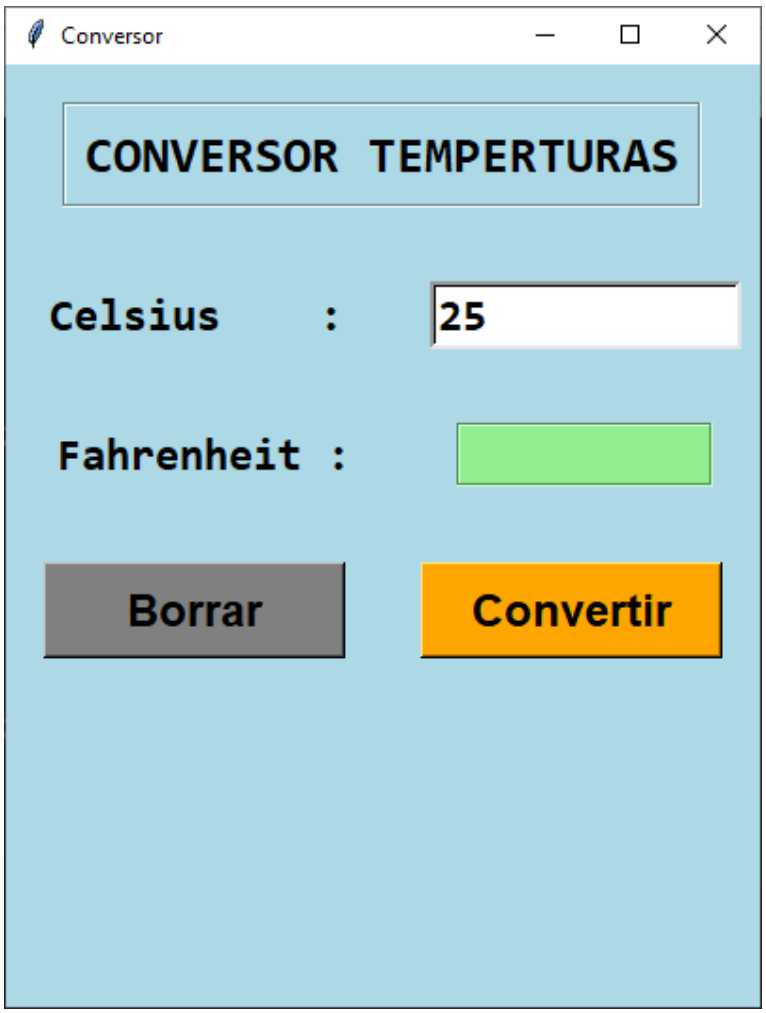
Hemos realizado las siguientes modificaciones:

```
47    ##### FRAME SEGUNDO #####
48    cuadro2 = tk.Frame(ventana,
49        bg="lightblue")
50    rotulo_fahrenheit = tk.Label(cuadro2,
51        text="Fahrenheit : ",
52        bg="lightblue",
53        font="consolas 18 bold")
54    rotulo_fahrenheit.pack(side=tk.LEFT, padx=20, pady=20)
55    salida_fahrenheit = tk.Label(cuadro2,
56        text="",
57        bg="lightgreen",
58        relief="groove",
59        font="consolas 18 bold",
60        width=10,
61        anchor=tk.E)
62    salida_fahrenheit.pack(side=tk.LEFT, padx=20, pady=20)
63    cuadro2.pack(pady=10)
```

En el marco cuadro1 vamos a cambiar el rótulo por una entrada de texto.

```
18  ##### FRAME PRIMERO #####
19  cuadro1 = tk.Frame(ventana,
20      bg="lightblue")
21  rotulo_celsius = tk.Label(cuadro1,
22      text="Celsius    : ",
23      bg="lightblue",
24      font="consolas 18 bold")
25  rotulo_celsius.pack(side=tk.LEFT, padx=20, pady=20)
26  entrada_grados= tk.Entry(cuadro1,
27      bg="white",fg="black",
28      font="consolas 18 bold",
29      relief=tk.SUNKEN, bd=3)
30  entrada_grados.pack(side=tk.LEFT, padx=10, pady=10)
31  cuadro1.pack()
```

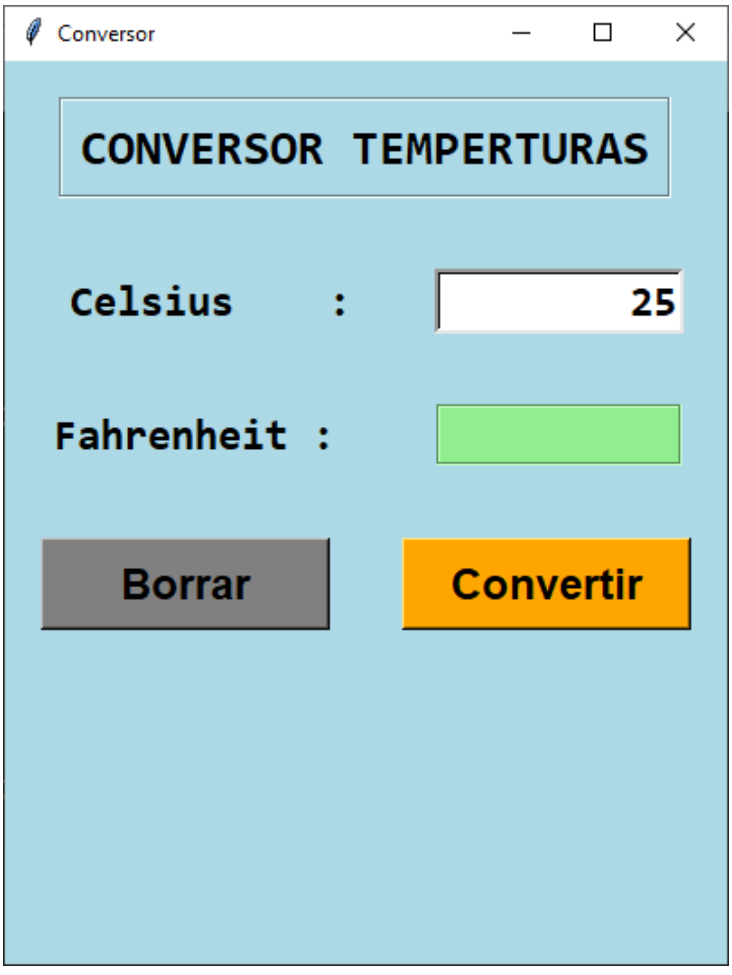
Hemos eliminado el rotulo para cambiarlo a una entrada de texto.



Ya podemos introducir números.

```
18   ##### FRAME PRIMERO #####
19   cuadro1 = tk.Frame(ventana,
20       bg="lightblue")
21   rotulo_celsius = tk.Label(cuadro1,
22       text="Celsius    : ",
23       bg="lightblue",
24       font="consolas 18 bold")
25   rotulo_celsius.pack(side=tk.LEFT, padx=20, pady=20)
26   entrada_grados= tk.Entry(cuadro1,
27       bg="white",fg="black",
28       font="consolas 18 bold",
29       relief=tk.SUNKEN, bd=3,
30       width=10,
31       justify=tk.RIGHT)
32   entrada_grados.pack(side=tk.LEFT, padx=10, pady=10)
33   cuadro1.pack()
```

El tamaño de la entrada en a 10 caracteres y la entrada de números será por la derecha.



9.- Método get para las cajas de texto o widgets Entry

Siguiendo con el proyecto anterior de conversor temperaturas, nos faltaba la funcionalidad.

Para recoger la información introducida por el usuario de una caja de texto vamos a utilizar el método get.

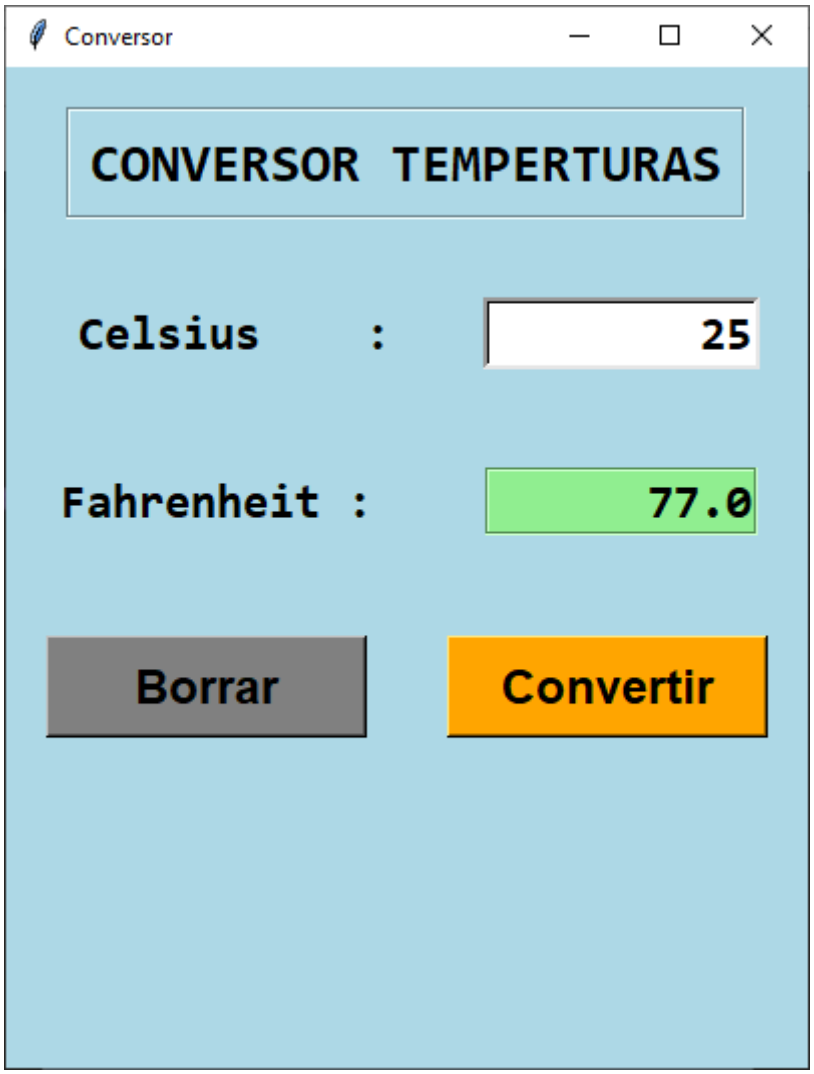
Vamos a crear la función para convertir grados a Fahrenheit.

```
 9    ######## FUNCIONES ########
10    def convertir():
11        try:
12            grados = entrada_grados.get()
13            resultado = (float(grados)* 1.8) + 32
14            salida_fahrenheit.config(text=resultado)
15        except ValueError:
16            pass
```

Al hacer clic en el botón convertir llama a la función.

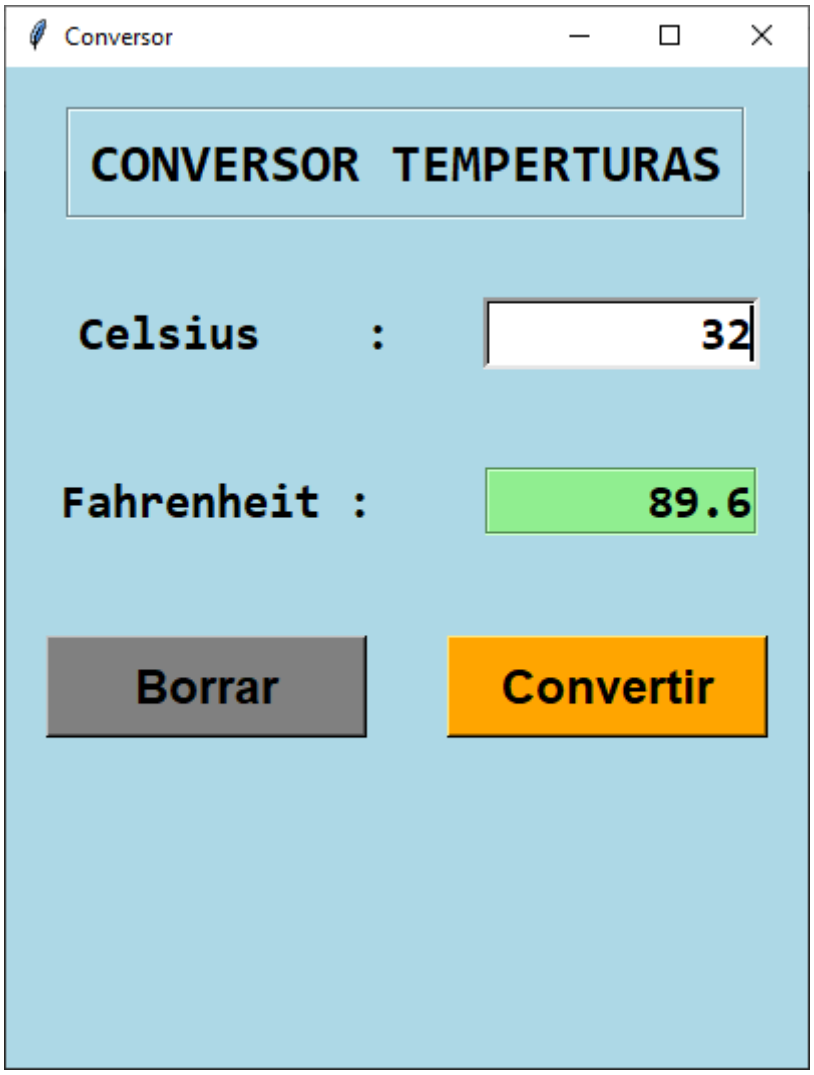
```
63    ##### FRAME TERCERO #####
64    cuadro3 = tk.Frame(ventana,
65        bg="lightblue")
66    boton_borrar = tk.Button(cuadro3,
67        text="Borrar",
68        bg="grey",
69        font="consolas, 18 bold",
70        width=10)
71    boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
72    boton_convertir = tk.Button(cuadro3,
73        text="Convertir",
74        bg="orange",
75        font="consolas, 18 bold",
76        width=10, command=convertir)
77    boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
78    cuadro3.pack()
```

Vamos a ejecutar el programa e introduciremos 25 grados Celsius para que nos diga los grados Fahrenheit.


Conversor
CONVERSOR TEMPERTURAS
Celsius :
25
Fahrenheit :
77.0
Borrar
Convertir

10.- Deshabilitar widgets

Continuando con el proyecto anterior.



Para evitar posibles confusiones si se ha dado al botón convertir o no, vamos a deshabilitar la caja de texto y el botón convertir hasta que no le demos al botón Borrar.

No poder introducir ningún dato más hasta que no le demos a Borrar.

```
30  ##### FRAME PRIMERO #####
31  cuadro1 = tk.Frame(ventana,
32      bg="lightblue")
33  rotulo_celsius = tk.Label(cuadro1,
34      text="Celsius    : ",
35      bg="lightblue",
36      font="consolas 18 bold")
37  rotulo_celsius.pack(side=tk.LEFT, padx=20, pady=20)
```

```
37    rotulo_celsius.pack(side=tk.LEFT, padx=20, pady=20)
38    entrada_grados= tk.Entry(cuadro1,
39        bg="white",fg="black",
40        font="consolas 18 bold",
41        relief=tk.SUNKEN, bd=3,
42        width=10,
43        justify=tk.RIGHT,
44        state="normal")
45    entrada_grados.pack(side=tk.LEFT, padx=10, pady=10)
46    cuadro1.pack()
```

En el objeto rotulo_celsius hemos agregado otra propiedad que es él state esta puede ser normal o disabled. En disabled no podremos introducir datos.

```
66    ##### FRAME TERCERO #####
67    cuadro3 = tk.Frame(ventana,
68        bg="lightblue")
69    boton_borrar = tk.Button(cuadro3,
70        text="Borrar",
71        bg="grey",
72        font="consolas, 18 bold",
73        width=10)
74    boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
75    boton_convertir = tk.Button(cuadro3,
76        text="Convertir",
77        bg="orange",
78        font="consolas, 18 bold",
79        width=10, command=convertir,
80        state="normal")
81    boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
82    cuadro3.pack()
```

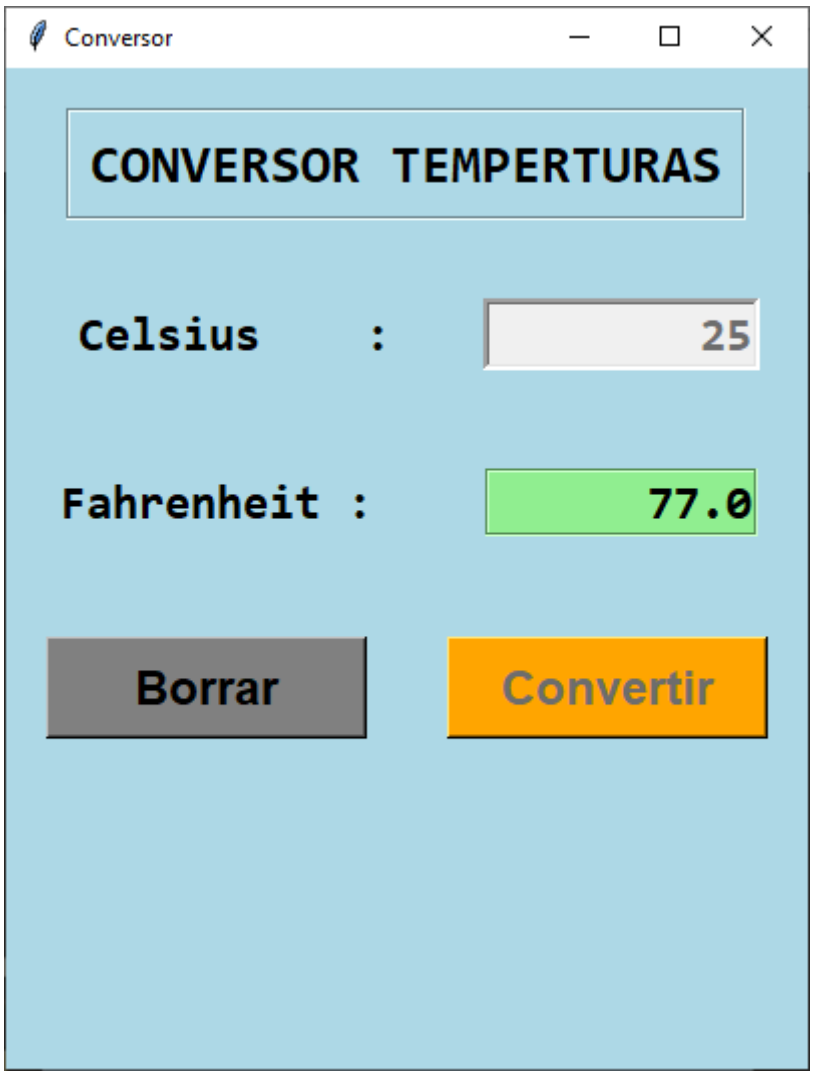
Lo mismo lo hacemos con el botón_convertir si state es igual a disabled no reaccionará el botón al hacer clic.

Vamos a modificar la función convertir:

```
 9    ######## FUNCIONES ########
10    def convertir():
11        try:
12            grados = entrada_grados.get()
13            resultado = (float(grados)* 1.8) + 32
```

```
14            salida_fahrenheit.config(text=resultado)
15            entrada_grados["state"] = "disabled"
16            boton_convertir["state"] = "disabled"
17        except ValueError:
18            pass
```

No podremos introducir datos y el botón_convertir no estará operativo.




```
 9   ######## FUNCIONES ########
10  def convertir():
11      try:
12          grados = entrada_grados.get()
13          resultado = (float(grados)* 1.8) + 32
14          salida_fahrenheit.config(text=resultado)
15          entrada_grados.config(state="disabled")
```

```
            boton_convertir.config(state="disabled")
        except ValueError:
            pass
```

Es otra forma de hacerlo.

Vamos a definir una segunda función para el botón Borrar.

```
    def borrar():
        entrada_grados.config(state="normal")
        boton_convertir.config(state="normal")
        entrada_grados.delete(0, tk.END)
        salida_fahrenheit.config(text="")
```

Ahora se la vamos a asignar.

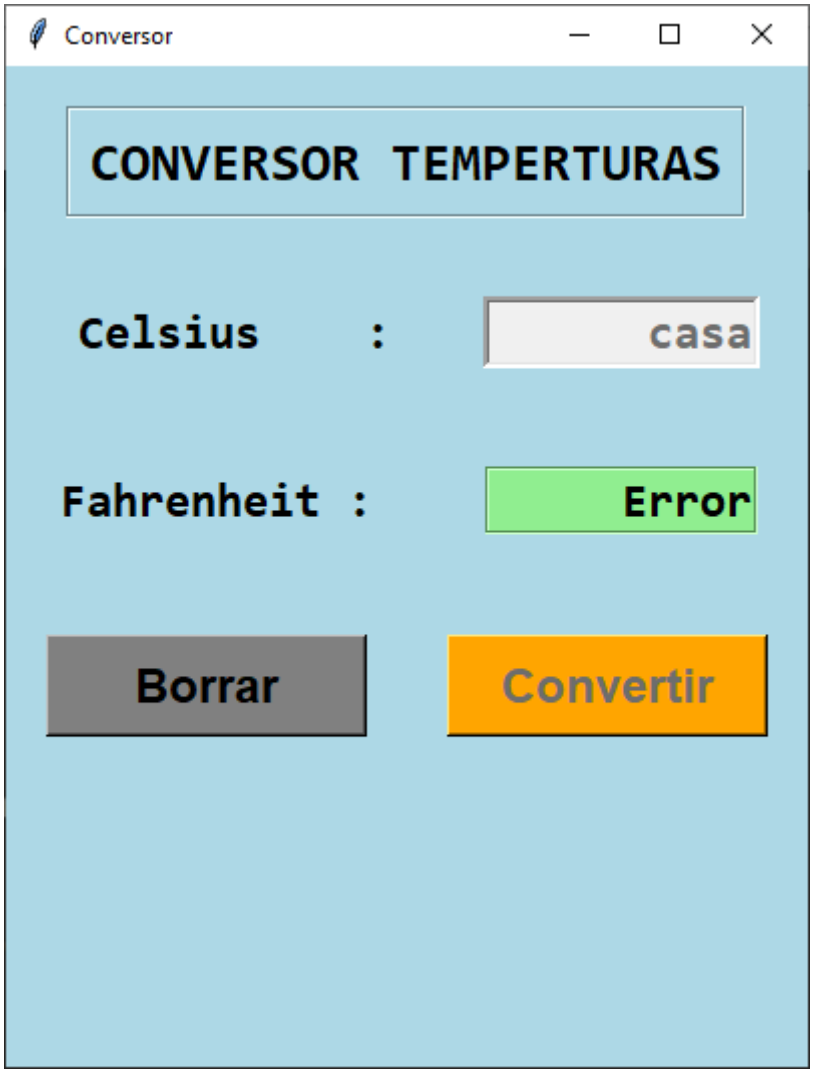
```
##### FRAME TERCERO #####
cuadro3 = tk.Frame(ventana,
    bg="lightblue")
boton_borrar = tk.Button(cuadro3,
    text="Borrar",
    bg="grey",
    font="consolas, 18 bold",
    width=10,command=borrar)
boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
boton_convertir = tk.Button(cuadro3,
    text="Convertir",
    bg="orange",
    font="consolas, 18 bold",
    width=10, command=convertir,
    state="normal")
boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
cuadro3.pack()
```

Cuando le damos al botón borrar se pone en modo normal tanto la entrada de texto como el botón Convertir, además de borrar el contenido de la entrada de Celsius como la etiqueta Fahrenheit.

Queremos que nos envíe un mensaje de error en la etiqueta Fahrenheit.

Vamos a modificar la primera función.

```
10  def convertir():
11      try:
12          grados = entrada_grados.get()
13          resultado = (float(grados)* 1.8) + 32
14          salida_fahrenheit.config(text=resultado)
15      except ValueError:
16          salida_fahrenheit.config(text="Error")
17      finally:
18          entrada_grados.config(state="disabled")
19          boton_convertir.config(state="disabled")
```

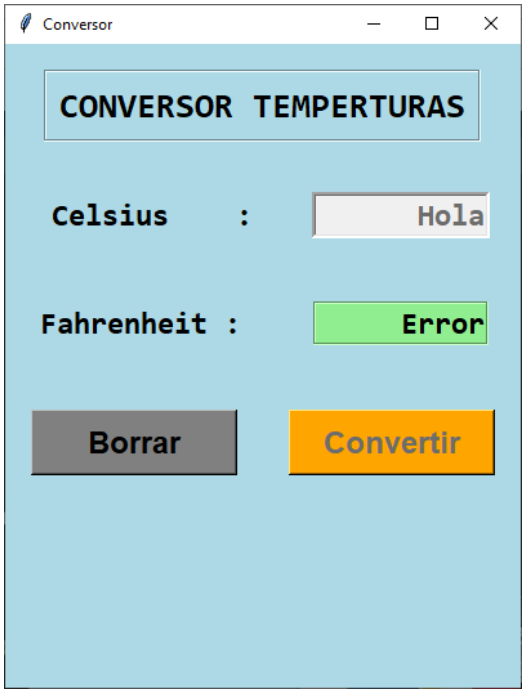


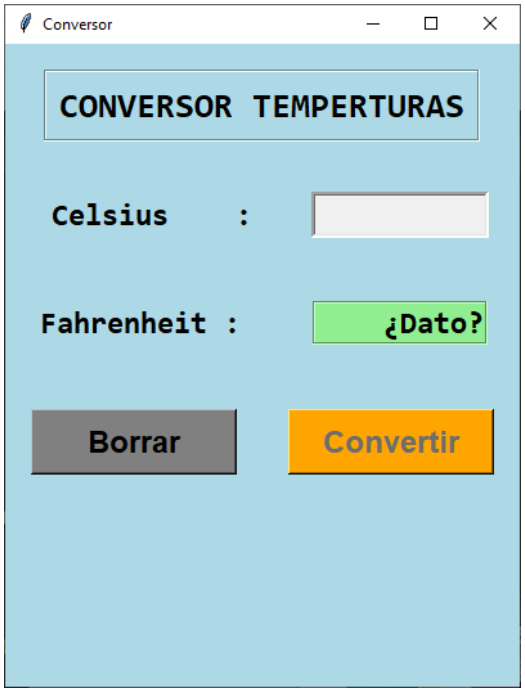
Tanto si hay error como si no lo hay la opción finally se ejecutará.

```
 9  ######## FUNCIONES ########
10  def convertir():
11      try:
```

```
12          grados = entrada_grados.get()
13          resultado = (float(grados)* 1.8) + 32
14          salida_fahrenheit.config(text=resultado)
15      except ValueError:
16          if grados == "":
17              salida_fahrenheit.config(text="¿Dato?")
18          else:
19              salida_fahrenheit.config(text="Error")
20      finally:
21          entrada_grados.config(state="disabled")
22          boton_convertir.config(state="disabled")
```

Si le damos al botón convertir y la caja de texto Celsius nos diga como mensaje de error ¿Dato?, pero si lo que hemos introducido es texto en lugar de números que nos diga Error.



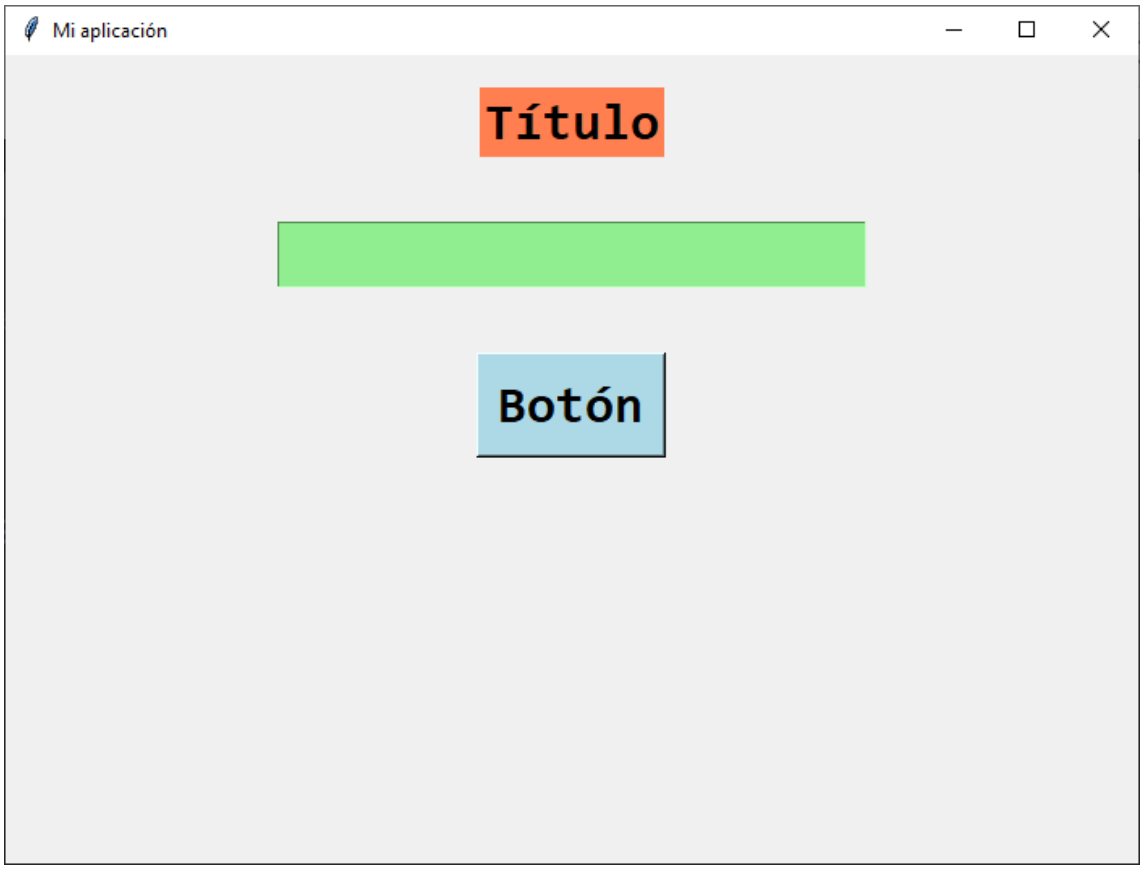


11.- Variables StringVar() en tkinter

Son las variables de control, que nos van a permitir más opciones que las variables que conocemos hasta ahora.

```
1   import tkinter as tk
2
3   ########## VENTANA ##########
4   ventana = tk.Tk()
5   ventana.title("Mi aplicación")
6   ventana.geometry("700x500+500+100")
7
8   ########## FUNCIONES ##########
9   def mostrar():
10      rotulo.config(text=entrada.get())
11
12  ########## ROTULO ###########
13  rotulo = tk.Label(ventana,
14      text="Título",
15      bg="coral",
16      font="consolas 24 bold")
17  rotulo.pack(padx=20, pady=20)
18
19  ######## ENTRADA ########
20  entrada = tk.Entry(ventana,
21      bg="lightgreen",
22      font="consolas 24 bold")
23  entrada.pack(padx=20, pady=20)
24
25  ######## BOTÓN #######
26  boton = tk.Button(ventana,
27      bg="lightblue",
28      text="Botón",
29      font="consolas 24 bold",
30      command=mostrar)
31  boton.pack(padx=20, pady=20)
```

```
32
33    ###### BUCLE PRINCIPAL ######
34    ventana.mainloop()
```



Vamos a ver como utilizar las variables de tkinter.

Las variables pueden se de tipo:

StringVar() ➔ Texto

IntVar() ➔ Números enteros

DoubleVar() ➔ Números con decimales (Float)

BooleanVar() ➔ de tipo boolean

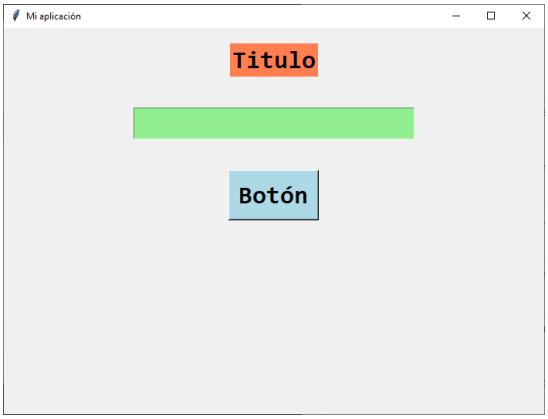
```
12    ########## ROTULO ###########
13    titulo = tk.StringVar()
14
15    rotulo = tk.Label(ventana,
16        bg="coral",
17        font="consolas 24 bold",
```

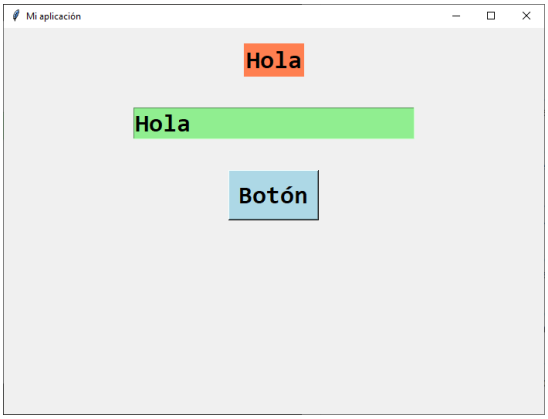
```
18        textvariable=titulo)
19    rotulo.pack(padx=20, pady=20)
20
21    titulo.set("Titulo")
```

En el apartado de ROTULO realiza los correspondientes cambios.

Vamos a modificar el apartado FUNCIONES.

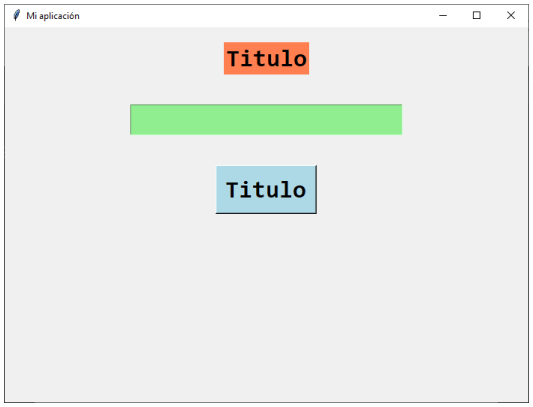
```
 8    ########## FUNCIONES ##########
 9    def mostrar():
10        titulo.set(entrada.get())
```

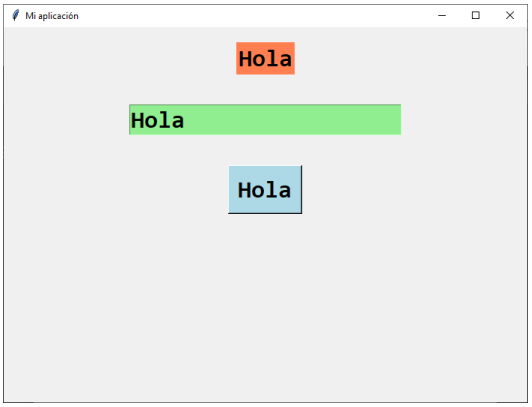




En el apartado de BOTÓN realizamos el siguiente cambio.

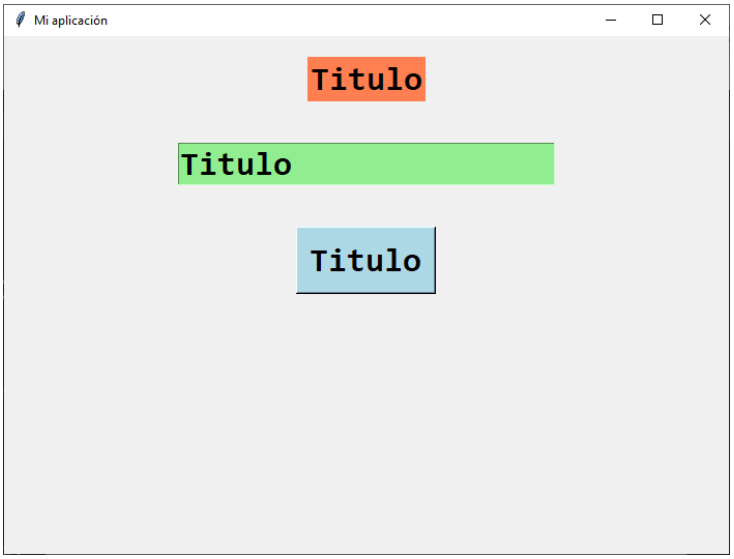
```
29    ######## BOTÓN ########
30    boton = tk.Button(ventana,
31        bg="lightblue",
32        textvariable=titulo,
33        font="consolas 24 bold",
34        command=mostrar)
35    boton.pack(padx=20, pady=20)
```

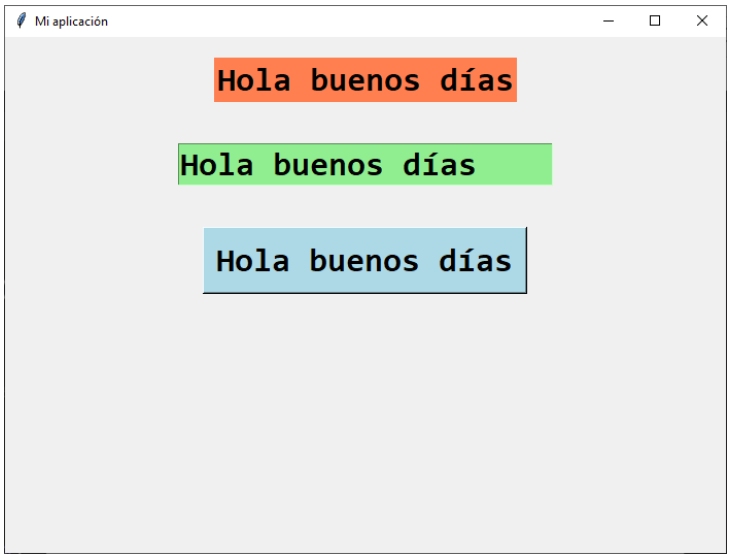




En el apartado ENTRADA realizamos el correspondiente cambio.

```
23    ######## ENTRADA ########
24    entrada = tk.Entry(ventana,
25        bg="lightgreen",
26        font="consolas 24 bold",
27        textvariable=titulo)
28    entrada.pack(padx=20, pady=20)
```

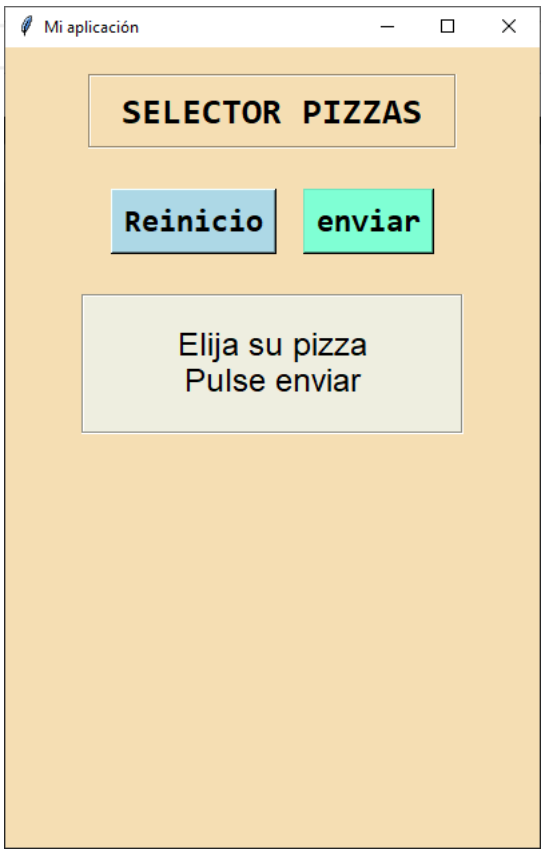




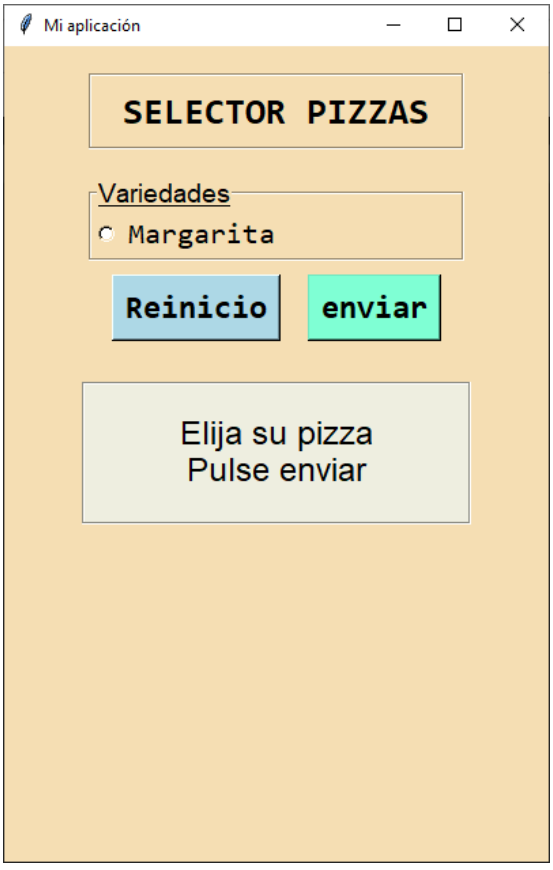
12.- Widget Radiobutton en tkinter

```
1   import tkinter as tk
2
3   ventana = tk.Tk()
4   ventana.title("Mi aplicación")
5   ventana.geometry("400x500+700+100")
6   ventana.minsize(width=400, height=600)
7   ventana.configure(bg="wheat")
8
9   ##### FUNCIONES #####
10  def enviar():
11      pass
12
13  def reiniciar():
14      pass
15
16  ##### ROTULO DEL TITULO #####
17  rotulo_titulo = tk.Label(ventana,
18      text="SELECTOR PIZZAS",
19      bg="wheat", fg="black",
20      font="consolas 20 bold",
21      relief = tk.GROOVE, bd=2,
22      width=18, pady=10)
23  rotulo_titulo.pack(padx=20, pady=20)
24
25  ##### BOTONES DE OPCIONES #####
26
27
28  ##### CUADRO PARA BOTONES #####
29  cuadro2 = tk.Frame(ventana,
30      bg="wheat")
31  cuadro2.pack()
32
```

```
33
34  boton_reiniciar = tk.Button(cuadro2,
35      text="Reinicio",
36      font="consolas 18 bold",
37      bg="lightblue", fg="black")
38  boton_reiniciar.pack(side=tk.LEFT, padx=10, pady=10)
39
40  boton_enviar = tk.Button(cuadro2,
41      text="enviar",
42      font="consolas 18 bold",
43      bg="aquamarine", fg="black",
44      command=enviar)
45  boton_enviar.pack(side=tk.LEFT, padx=10, pady=10)
46
47  rotulo_resultado = tk.Label(ventana,
48      text="Elija su pizza\nPulse enviar",
49      bg="ivory2", fg="black",
50      font="arial 18",
51      anchor=tk.CENTER,
52      relief=tk.GROOVE, bd=2,
53      width=20, height=3,
54      pady=10)
55  rotulo_resultado.pack(padx=20, pady=20)
56
57  ##### BUCLE PRINCIPAL #####
58  ventana.mainloop()
```



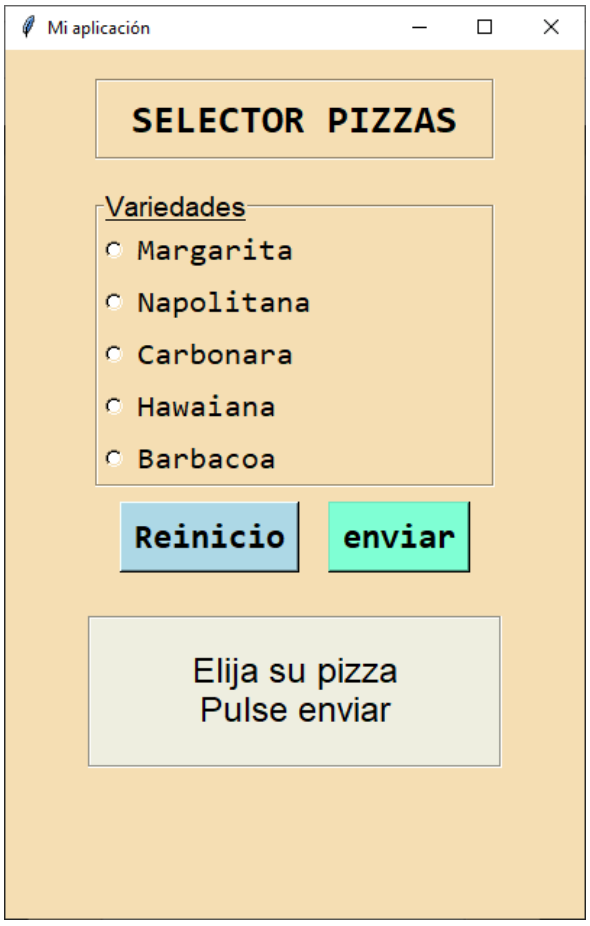
```
25  ##### BOTONES DE OPCIONES #####
26  cuadro1 = tk.LabelFrame(ventana,
27      bg="wheat",
28      text="Variedades",
29      font="arial 14 underline")
30
31  opcion_seleccionada = tk.IntVar()
32
33  opcion1 = tk.Radiobutton(cuadro1,
34      text = "Margarita",
35      bg="wheat",
36      font="consolas 16",
37      width=20, anchor=tk.W,
38      variable=opcion_seleccionada,
39      value=1)
40  opcion1.pack()
41
42  cuadro1.pack()
```



Completamos OPCIONES DE BOTONES:

```
25    ##### BOTONES DE OPCIONES #####
26    cuadro1 = tk.LabelFrame(ventana,
27        bg="wheat",
28        text="Variedades",
29        font="arial 14 underline")
30
31    opcion_seleccionada = tk.IntVar()
32
33    opcion1 = tk.Radiobutton(cuadro1,
34        text = "Margarita",
35        bg="wheat",
36        font="consolas 16",
37        width=20, anchor=tk.W,
38        variable=opcion_seleccionada,
39        value=1)
40    opcion1.pack()
41
42    opcion2 = tk.Radiobutton(cuadro1,
43        text = "Napolitana",
44        bg="wheat",
45        font="consolas 16",
46        width=20, anchor=tk.W,
47        variable=opcion_seleccionada,
48        value=2)
49    opcion2.pack()
50
51    opcion3 = tk.Radiobutton(cuadro1,
52        text = "Carbonara",
53        bg="wheat",
54        font="consolas 16",
55        width=20, anchor=tk.W,
56        variable=opcion_seleccionada,
57        value=3)
```

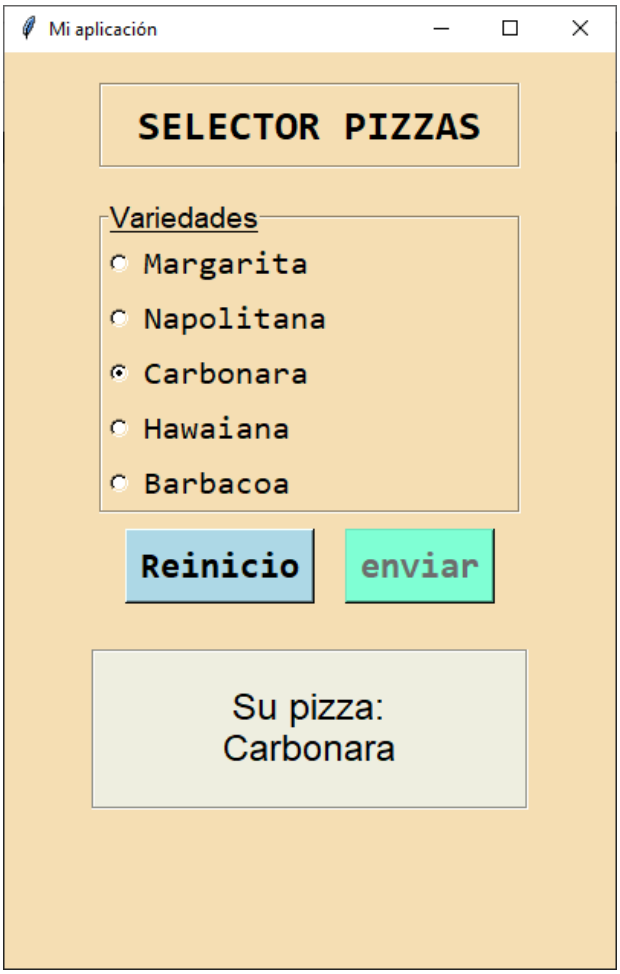
```
58    opcion3.pack()
59
60    opcion4 = tk.Radiobutton(cuadro1,
61        text = "Hawaiana",
62        bg="wheat",
63        font="consolas 16",
64        width=20, anchor=tk.W,
65        variable=opcion_seleccionada,
66        value=4)
67    opcion4.pack()
68
69    opcion5 = tk.Radiobutton(cuadro1,
70        text = "Barbacoa",
71        bg="wheat",
72        font="consolas 16",
73        width=20, anchor=tk.W,
74        variable=opcion_seleccionada,
75        value=5)
76    opcion5.pack()
77
78    cuadro1.pack()
```



13.- Funcionalidad a botones de opciones o radio botones

Vamos a redefinir las funciones:

```
10    def enviar():
11        pizza_elegida = opcion_seleccionada.get()
12        if pizza_elegida == 1:
13            resultado = "Margarita"
14        elif pizza_elegida ==2:
15            resultado = "Napolitana"
16        elif pizza_elegida ==3:
17            resultado = "Carbonara"
18        elif pizza_elegida ==4:
19            resultado = "Hawaiana"
20        elif pizza_elegida ==5:
21            resultado = "Barbacoa"
22        else:
23            resultado = "No hay selección"
24        rotulo_resultado.config(text=f"Su pizza:\n{resultado}")
25        boton_enviar['state']='disabled'
26        boton_reiniciar['state']='normal'
```



Ya podemos selecciona pizza y dar al botón enviar, además de mostrarse en la etiqueta inferior, en botón enviar queda deshabilitado para que no puedas presionarlo hasta que presiones el botón Reinicio.
Vamos a definir la segunda función.

```
28  def reiniciar():
29      opcion_seleccionada.set(0)
30      rotulo_resultado.config(text="Elija su pizza\nPulse enviar")
31      boton_enviar['state']='normal'
32      boton_reiniciar['state']='disabled'
```

Ahora vamos a asignar esta función a su correspondiente botón.

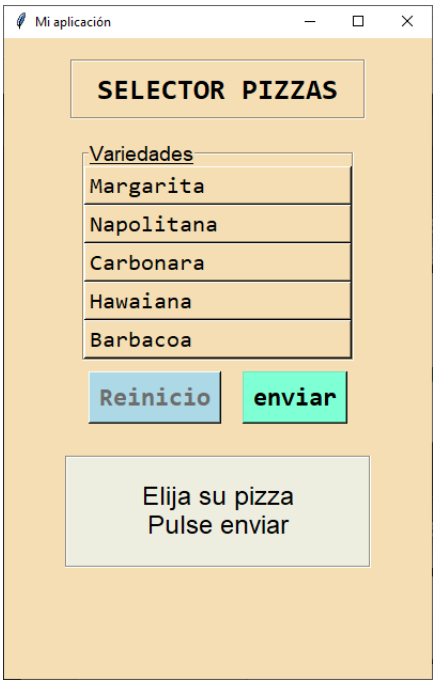
```
103   boton_reiniciar = tk.Button(cuadro2,
104       text="Reinicio",
105       font="consolas 18 bold",
106       bg="lightblue", fg="black",
107       command=reiniciar)
108   boton_reiniciar.pack(side=tk.LEFT, padx=10, pady=10)
```

Ahora como curiosidad:

```
51   opcion1 = tk.Radiobutton(cuadro1,
52       text = "Margarita",
53       bg="wheat",
54       font="consolas 16",
55       width=20, anchor=tk.W,
56       indicator=0,
57       variable=opcion_seleccionada,
58       value=1)
59   opcion1.pack()
```

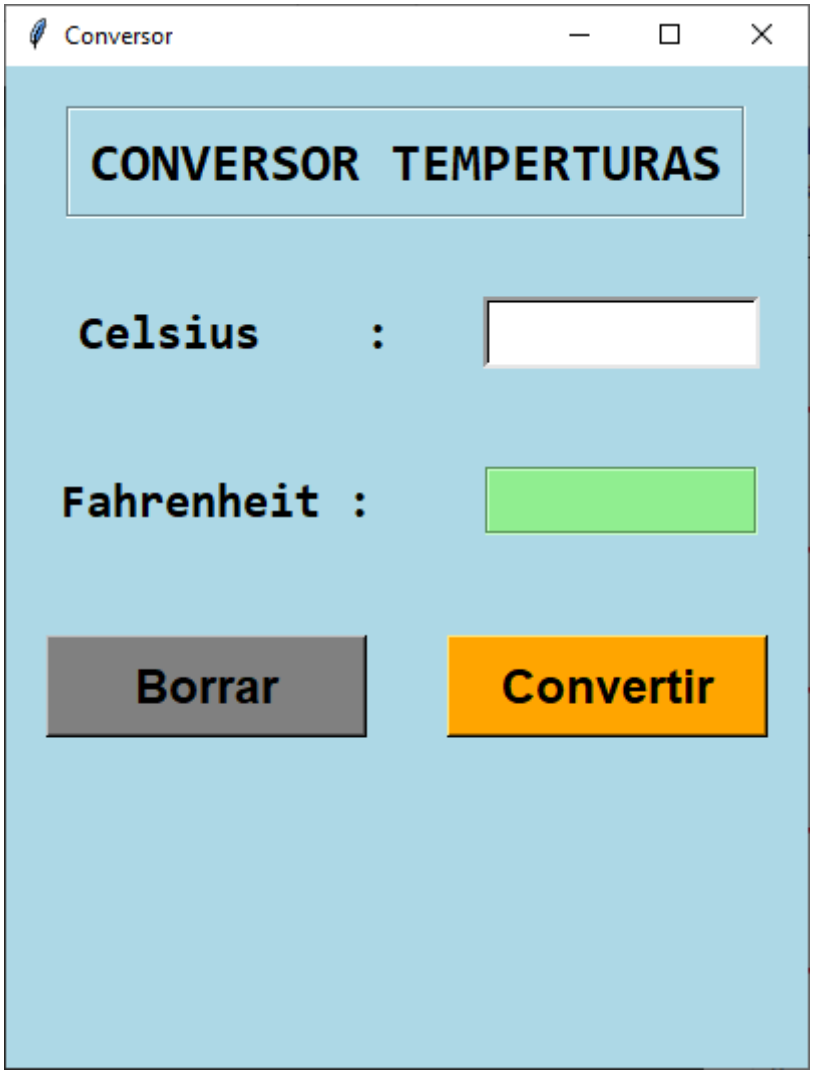
A todos los Radiobutton le agregamos la siguiente propiedad (indicator=0).

Ahora ha cambiado a botones.



14.- Opciones command con los radio botones

Volvemos de nuevo al proyecto "CONVERSOR TEMPERTURAS" para realizar unas modificaciones.

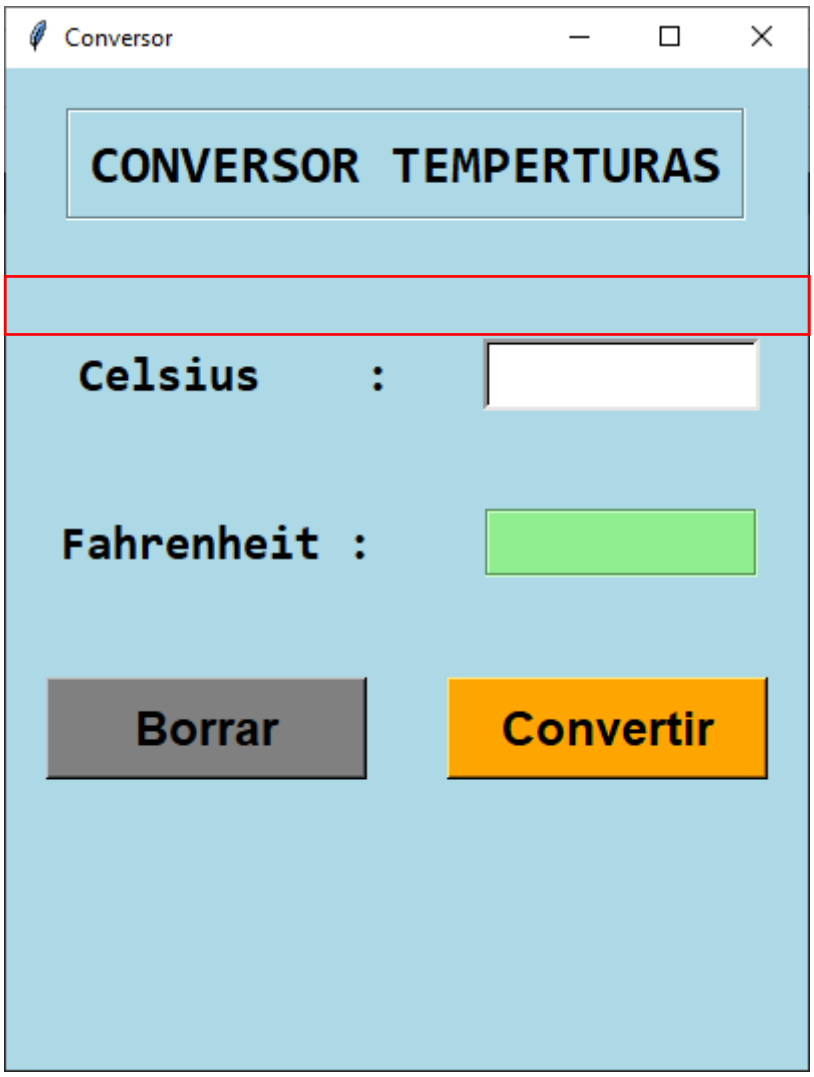


Vamos a agregar dos radio botones par poder seleccionar si deseamos convertir de Celsius a Fahrenheit o de Fahrenheit a Celsius.

```
35    ##### LABEL TITULO #####
36    rotulo_titulo = tk.Label(ventana,
37        text="CONVERSOR TEMPERTURAS",
38        bg="lightblue", fg="black",
39        font= "consolas 20 bold",
40        relief=tk.GROOVE, bd=2,
41        padx=10, pady=10)
42    rotulo_titulo.pack(padx=20, pady=20)
```

```
43
44    ##### CUADRO ESPACIO #####
45    cuadro_espacio = tk.Frame(ventana,
46        height=20,
47        bg="lightblue")
48    cuadro_espacio.pack()
49
50    ##### FRAME PRIMERO #####
51    cuadro1 = tk.Frame(ventana,
52        bg="lightblue")
```

Entre LABEL TITULO y FRAME PRIMERO hemos agregado CUADRO ESPACIO, esto nos permitirá que haga una separación entre el título de proyecto y el cuadro1 que contiene la etiqueta Celsius más la caja de texto.



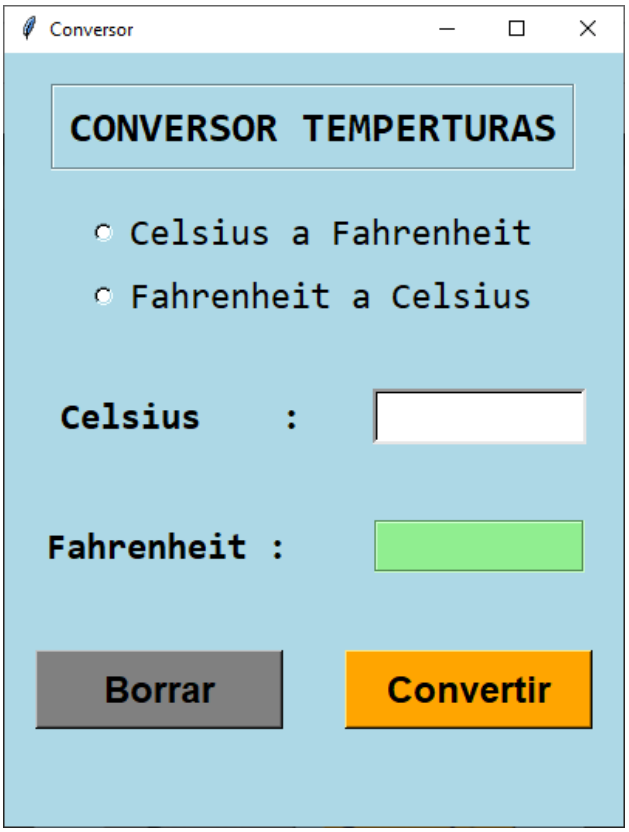
Antes de CUADRO ESPACIO agregaremos el siguiente código:

```
45    ###### BOTONES DE OPCIONES ######
46    opcion_conversion = tk.IntVar()
47    opcion1 = tk.Radiobutton(ventana,
48        text="Celsius a Fahrenheit",
49        bg="lightblue",
50        font="consolas 18",
51        variable=opcion_conversion,
52        value=1,
53        command=cambiar)
54    opcion1.pack()
55
56    opcion2 = tk.Radiobutton(ventana,
57        text="Fahrenheit a Celsius",
58        bg="lightblue",
59        font="consolas 18",
60        variable=opcion_conversion,
61        value=2,
62        command=cambiar)
63    opcion2.pack()
```

Hemos creado la función sin definir el código:

```
32    def cambiar():
33        pass
```

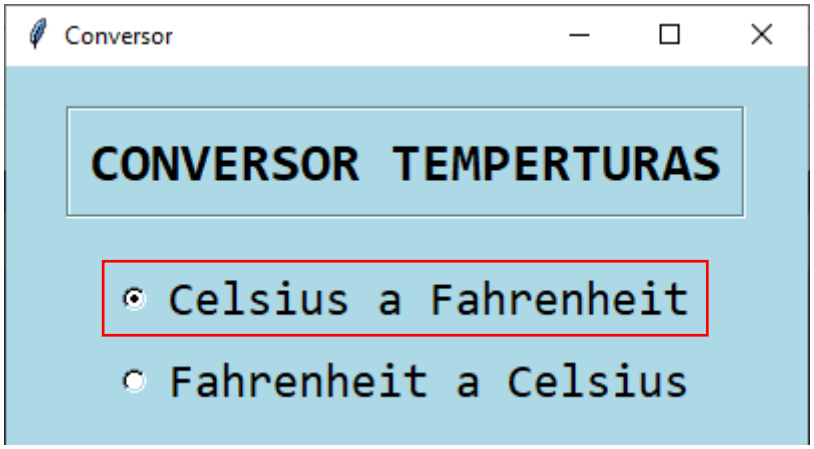
Esto nos permitirá probar el código sin posible errores.



Al final de BOTONES OPCIONES pondremos el siguiente código:

```
64    opcion_conversion.set(1)
```

Esto nos permitirá que la primera opción de los radio botón este activada.



Ya que es la reconversión que se suele realizar más veces.

Vamos a definir la función cambiar.

```
##### FRAME PRIMERO #####
cuadro1 = tk.Frame(ventana,
    bg="lightblue")
rotulo_primero = tk.Label(cuadro1,
    text="Celsius    : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_primero.pack(side=tk.LEFT, padx=20, pady=20)
```

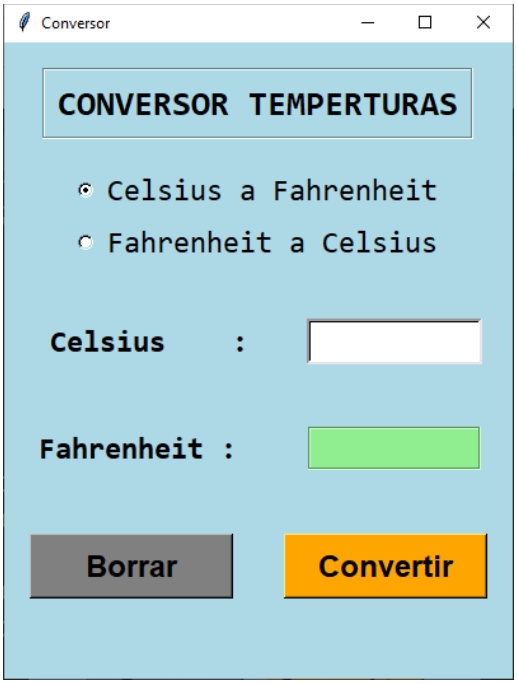
```
93    ##### FRAME SEGUNDO #####
94    cuadro2 = tk.Frame(ventana,
95        bg="lightblue")
96    rotulo_segundo = tk.Label(cuadro2,
97        text="Fahrenheit : ",
98        bg="lightblue",
99        font="consolas 18 bold")
100   rotulo_segundo.pack(side=tk.LEFT, padx=20, pady=20)
```

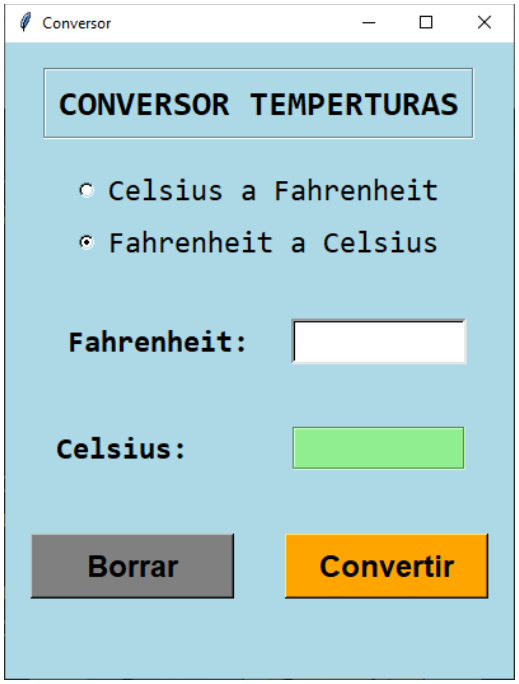
Hemos modificado el nombre de las etiquetas que identifican el tipo de grados.

Agregamos el código a la función cambiar()

```
32    def cambiar():
33        if opcion_conversion.get()== 1:
34            rotulo_primero.config(text="Celsius:    ")
35            rotulo_segundo.config(text="Fahrenheit:")
36        else:
37            rotulo_primero.config(text="Fahrenheit:")
38            rotulo_segundo.config(text="Celsius:    ")
```

Si ahora lo probamos:





Vamos a cambiar la función convertir:

Cambia el nombre del objeto salida_Fahrenheit por rotulo_resultado.

Te adjunto todo el código para poder consultar algunos errores que hayas podido tener al realizar los pasos de este tutorial:

```
import tkinter as tk

ventana = tk.Tk()
ventana.title("Conversor")
ventana.geometry("400x500+700+100")
ventana.minsize(width=400, height=500)
ventana.configure(bg="lightblue")

######## FUNCIONES ########
def convertir():
    try:
        grados = entrada_grados.get()
```

```
            if opcion_conversion.get() == 1:
                resultado = round(float(grados)* 1.8) + 32
            else:
                resultado = round((float(grados)-32)/1.8)
            rotulo_resultado.config(text=resultado)
        except ValueError:
            if grados == "":
                rotulo_resultado.config(text="¿Dato?")
            else:
                rotulo_resultado.config(text="Error")
        finally:
            entrada_grados.config(state="disabled")
            boton_convertir.config(state="disabled")



def borrar():
    entrada_grados.config(state="normal")
    boton_convertir.config(state="normal")
    entrada_grados.delete(0, tk.END)
    rotulo_resultado.config(text="")

def cambiar():
    if opcion_conversion.get()== 1:
        rotulo_primero.config(text="Celsius:   ")
        rotulo_segundo.config(text="Fahrenheit:")
    else:
        rotulo_primero.config(text="Fahrenheit:")
        rotulo_segundo.config(text="Celsius:   ")



##### LABEL TITULO #####
rotulo_titulo = tk.Label(ventana,
    text="CONVERSOR TEMPERTURAS",
    bg="lightblue", fg="black",
    font= "consolas 20 bold",
    relief=tk.GROOVE, bd=2,
    padx=10, pady=10)
rotulo_titulo.pack(padx=20, pady=20)

###### BOTONES DE OPCIONES ######
opcion_conversion = tk.IntVar()
opcion1 = tk.Radiobutton(ventana,
    text="Celsius a Fahrenheit",
    bg="lightblue",
    font="consolas 18",
    variable=opcion_conversion,
    value=1,
```

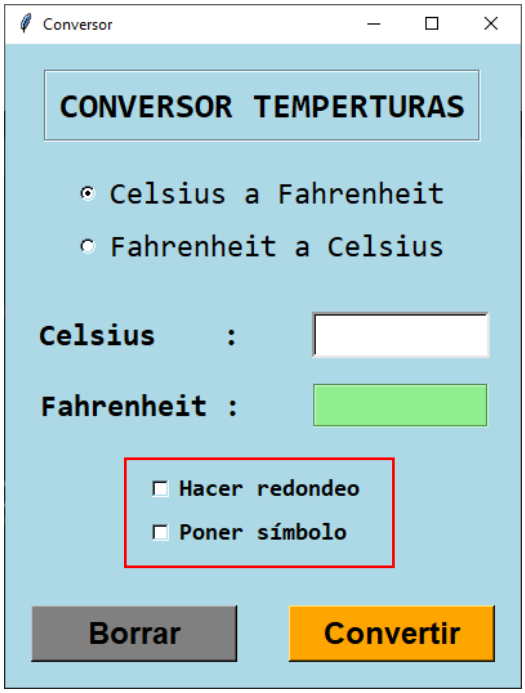
```
    command=cambiar)
opcion1.pack()

opcion2 = tk.Radiobutton(ventana,
    text="Fahrenheit a Celsius",
    bg="lightblue",
    font="consolas 18",
    variable=opcion_conversion,
    value=2,
    command=cambiar)
opcion2.pack()
opcion_conversion.set(1)

##### CUADRO ESPACIO #####
cuadro_espacio = tk.Frame(ventana,
    height=20,
    bg="lightblue")
cuadro_espacio.pack()

##### FRAME PRIMERO #####
cuadro1 = tk.Frame(ventana,
    bg="lightblue")
rotulo_primero = tk.Label(cuadro1,
    text="Celsius    : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_primero.pack(side=tk.LEFT, padx=20, pady=20)
entrada_grados= tk.Entry(cuadro1,
    bg="white",fg="black",
    font="consolas 18 bold",
    relief=tk.SUNKEN, bd=3,
    width=10,
    justify=tk.RIGHT,
    state="normal")
entrada_grados.pack(side=tk.LEFT, padx=20, pady=20)
cuadro1.pack(pady=10)

##### FRAME SEGUNDO #####
cuadro2 = tk.Frame(ventana,
    bg="lightblue")
rotulo_segundo = tk.Label(cuadro2,
    text="Fahrenheit : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_segundo.pack(side=tk.LEFT, padx=20, pady=20)
rotulo_resultado = tk.Label(cuadro2,
    text="",
    bg="lightgreen",
    relief="groove",
```

```
    font="consolas 18 bold",
    width=10,
    anchor=tk.E)
rotulo_resultado.pack(side=tk.LEFT, padx=20, pady=20)
cuadro2.pack(pady=10)

##### FRAME TERCERO #####
cuadro3 = tk.Frame(ventana,
    bg="lightblue")
boton_borrar = tk.Button(cuadro3,
    text="Borrar",
    bg="grey",
    font="consolas, 18 bold",
    width=10,command=borrar)
boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
boton_convertir = tk.Button(cuadro3,
    text="Convertir",
    bg="orange",
    font="consolas, 18 bold",
    width=10, command=convertir,
    state="normal")
boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
cuadro3.pack()

ventana.mainloop()
```

15.- Widget Checkbutton

En este capítulo vamos a modifica de nuevo el CONVERSOR TEMPERATURAS añadiéndole 2 Checkbutton.

```
117    ##### CUADRO TERCERO #####
118    cuadro3 = tk.Frame(ventana,
119    bg = "lightblue")
120    redondeo = tk.BooleanVar()
121    casilla_redondeo = tk.Checkbutton(cuadro3,
122        text="Hacer redondeo",
123        bg="lightblue",
124        font="consolas 14 bold",
125        width=15, anchor=tk.W,
126        variable=redondeo)
127    casilla_redondeo.pack()
128
129    simbolo = tk.BooleanVar()
130    casilla_simbolo = tk.Checkbutton(cuadro3,
131        text="Poner símbolo",
132        bg="lightblue",
133        font="consolas 14 bold",
134        width=15, anchor=tk.W,
135        variable=simbolo)
136    casilla_simbolo.pack()
137
138    cuadro3.pack(pady=20)
```



Ya tenemos los Checkbutton.

Ahora le vamos a dar funcionalidad.

Modificamos la función convertir()

```
9   ######## FUNCIONES ########
10  def convertir():
11      try:
12          grados = entrada_grados.get()
13          if opcion_conversion.get() == 1:
14              resultado = (float(grados)* 1.8) + 32
15          else:
16              resultado = (float(grados)-32)/1.8
17
18          if redondeo.get():
19              resultado = round(resultado)
20          if simbolo.get():
21              if opcion_conversion.get() == 1:
22                  resultado = f"{resultado}\u00B0F"
23              else:
24                  resultado = f"{resultado}\u00B0C"
25          rotulo_resultado.config(text=resultado)
26      except ValueError:
27          if grados == "":
28              rotulo_resultado.config(text="¿Dato?")
29          else:
30              rotulo_resultado.config(text="Error")
31      finally:
32          entrada_grados.config(state="disabled")
33          boton_convertir.config(state="disabled")
```

Ahora te toca a ti a utilizar todas las combinaciones del CONVERSOR DE TEMPERATURA.

16.- El módulo ttk de tkinter

Antes de seguir con el siguiente capítulo te propongo que realices el siguiente diseño:

Para empezar a utilizar Ttk, hay que importar el módulo:

from tkinter import ttk.

Es necesario importarlos de esta forma para facilitar su uso.

Pero si lo importamos de esta otra manera:

from tkinter import *

from tkinter.ttk import *

Si un widget reemplazará automáticamente al widget clásico correspondiente en tkinter, por lo que tenemos que tener cuidado ya que como acabamos de decir la implementación de los widgets el módulo ttk no es igual en algunos aspectos a la de los widget de la librería tkinter directamente y puede dar lugar a errores.

Opciones como fg o bg ya no están presentes en ttk por lo que habría que codificarlos de forma diferente.

En algunos widgets no se va a notar la diferencia, en cambio en otro sí.

17.- Widget Combobox

```
1   import tkinter as tk
2   from tkinter import ttk
```

Tenemos que importar el ttk.

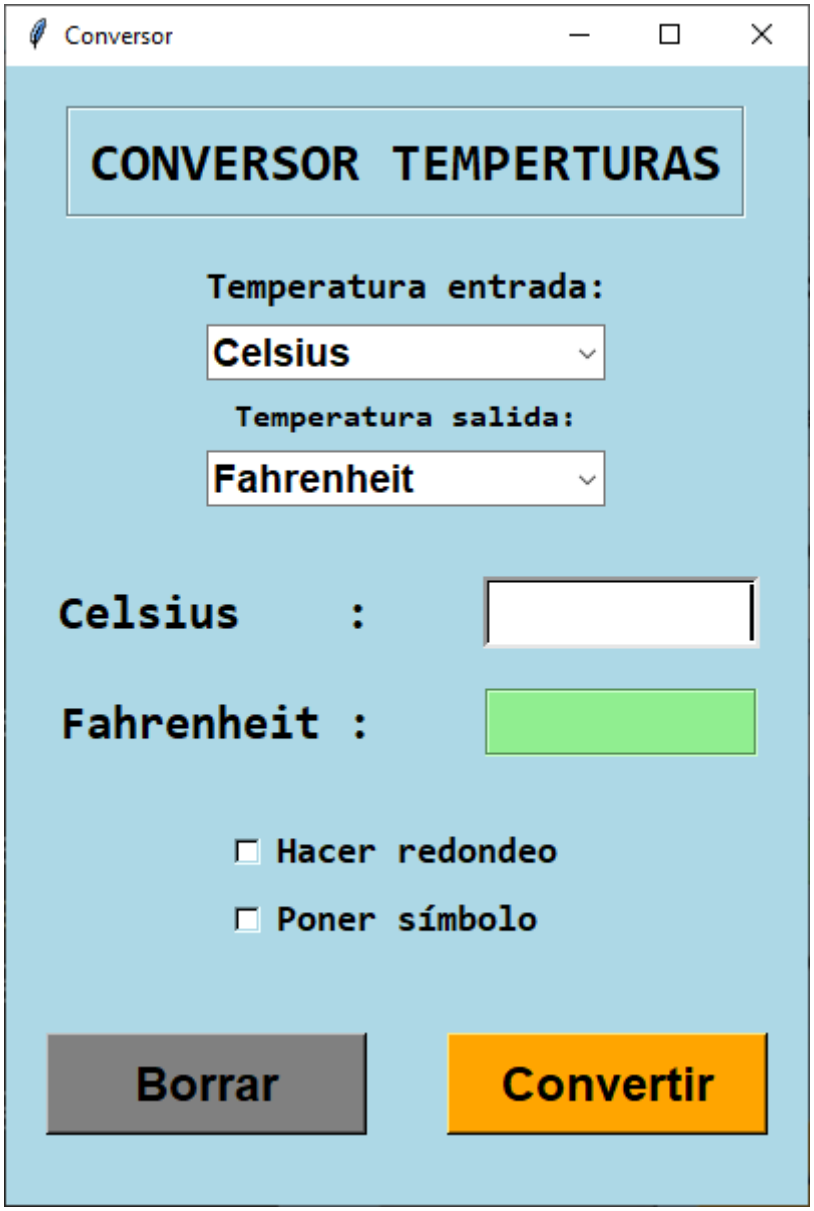
Creamos un nuevo bloque:

```
64  ###### DESPLEGABLES CON SUS RÓTULOS ######
65  rotulo_entrada = tk.Label(ventana,
66      text="Temperatura entrada:",
67      bg="lightblue", fg="black",
68      font="consolas 14 bold",
69      bd=2)
70  rotulo_entrada.pack()
71
72  temp_entrada = tk.StringVar()
73  desplegable_entrada = ttk.Combobox(ventana,
74      font="consola 14 bold",
75      width=16,
76      values=["Celsius", "Fahrenheit", "Kelvin"],
77      state="readonly",
78      textvariable=temp_entrada)
79  desplegable_entrada.pack(pady=5)
80  desplegable_entrada.set("Celsius")
81
82  rotulo_salida = tk.Label(ventana,
83      text="Temperatura salida:",
84      bg="lightblue", fg="black",
85      font="consolas 12 bold",
86      bd=2)
87  rotulo_salida.pack()
88   temp_salida = tk.StringVar()
89   desplegable_salida = ttk.Combobox(ventana,
90       font="consola 14 bold",
91       width=16,
92       values=["Celsius", "Fahrenheit", "Kelvin"],
93       state="readonly",
94       textvariable=temp_salida)
```

```
95    desplegable_salida.pack(pady=5)
96    desplegable_salida.set("Fahrenheit")
```

El apartado OPCIONES DE BOTONES lo borras o lo pasas a comentario:

“”” El código de varias líneas que se encuentre entre tres comillas dobles pasa a ser un comentario, a la hora de ejecutar este codigo será ignorado. “””.



Aquí tienes todo el código:

```
import tkinter as tk
from tkinter import ttk


ventana = tk.Tk()
ventana.title("Conversor")
ventana.geometry("400x500+700+100")
```

```
ventana.minsize(width=400, height=550)
ventana.configure(bg="lightblue")

######## FUNCIONES ########
def convertir():
    try:
        entrada = temp_entrada.get()
        salida = temp_salida.get()
        grados = float(entrada_grados.get())
        if entrada == salida:
            resultado = grados
        elif entrada == "Celsius" and salida == "Fahrenheit":
            resultado = round(grados * 1.8 +32, 2)
        elif entrada == "Fahrenheit" and salida == "Celsius":
            resultado = round((grados - 32)/ 1.8, 2)
        elif entrada == "Celsius" and salida == "Kelvin":
            resultado = round(grados + 273.15, 2)
        elif entrada == "Kelvin" and salida == "Celsius":
            resultado = round(grados - 273.15, 2)
        elif entrada == "fahrenheit" and salida == "Kelvin":
            resultado = round(((grados - 32)/1.8)+273.15, 2)

        if redondeo.get():
            resultado = round(resultado)
        if simbolo.get():
            if salida == "Celsius":
                resultado = f"{resultado}\u00B0C"
            elif salida == "Fahrenheit":
                resultado = f"{resultado}\u00B0F"
            else:
                resultado = f"{resultado} K"
        rotulo_resultado.config(text=resultado)
        entrada_grados.config(state="disabled")
        boton_convertir.config(state="disabled")
    except ValueError:
        if grados == "":
            rotulo_resultado.config(text="¿Dato?")
        else:
            rotulo_resultado.config(text="Error")
    finally:
        entrada_grados.config(state="disabled")
        boton_convertir.config(state="disabled")


def borrar():
    entrada_grados.config(state="normal")
    boton_convertir.config(state="normal")
    entrada_grados.delete(0, tk.END)
```

```
    rotulo_resultado.config(text="")

def cambiar():
    if opcion_conversion.get()== 1:
        rotulo_primero.config(text="Celsius:   ")
        rotulo_segundo.config(text="Fahrenheit:")
    else:
        rotulo_primero.config(text="Fahrenheit:")
        rotulo_segundo.config(text="Celsius:   ")



##### LABEL TITULO #####
rotulo_titulo = tk.Label(ventana,
    text="CONVERSOR TEMPERTURAS",
    bg="lightblue", fg="black",
    font= "consolas 20 bold",
    relief=tk.GROOVE, bd=2,
    padx=10, pady=10)
rotulo_titulo.pack(padx=20, pady=20)

###### DESPLEGABLES CON SUS RÓTULOS ######
rotulo_entrada = tk.Label(ventana,
    text="Temperatura entrada:",
    bg="lightblue", fg="black",
    font="consolas 14 bold",
    bd=2)
rotulo_entrada.pack()

temp_entrada = tk.StringVar()
desplegable_entrada = ttk.Combobox(ventana,
    font="consola 14 bold",
    width=16,
    values=["Celsius", "Fahrenheit", "Kelvin"],
    state="readonly",
    textvariable=temp_entrada)
desplegable_entrada.pack(pady=5)
desplegable_entrada.set("Celsius")

rotulo_salida = tk.Label(ventana,
    text="Temperatura salida:",
    bg="lightblue", fg="black",
    font="consolas 12 bold",
    bd=2)
rotulo_salida.pack()
temp_salida = tk.StringVar()
desplegable_salida = ttk.Combobox(ventana,
    font="consola 14 bold",
    width=16,
```

```
    values=["Celsius", "Fahrenheit", "Kelvin"],
    state="readonly",
    textvariable=temp_salida)
desplegable_salida.pack(pady=5)
desplegable_salida.set("Fahrenheit")


##### CUADRO ESPACIO #####
cuadro_espacio = tk.Frame(ventana,
    height=20,
    bg="lightblue")
cuadro_espacio.pack()

##### FRAME PRIMERO #####
cuadro1 = tk.Frame(ventana,
    bg="lightblue")
rotulo_primero = tk.Label(cuadro1,
    text="Celsius    : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_primero.pack(side=tk.LEFT, padx=20, pady=0)
entrada_grados= tk.Entry(cuadro1,
    bg="white",fg="black",
    font="consolas 18 bold",
    relief=tk.SUNKEN, bd=3,
    width=10,
    justify=tk.RIGHT,
    state="normal")
entrada_grados.pack(side=tk.LEFT, padx=20, pady=0)
cuadro1.pack(pady=10)

##### FRAME SEGUNDO #####
cuadro2 = tk.Frame(ventana,
    bg="lightblue")
rotulo_segundo = tk.Label(cuadro2,
    text="Fahrenheit : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_segundo.pack(side=tk.LEFT, padx=20, pady=0)
rotulo_resultado = tk.Label(cuadro2,
    text="",
    bg="lightgreen",
    relief="groove",
    font="consolas 18 bold",
    width=10,
    anchor=tk.E)
rotulo_resultado.pack(side=tk.LEFT, padx=20, pady=0)
cuadro2.pack(pady=10)
```

```
##### CUADRO TERCERO #####
cuadro3 = tk.Frame(ventana,
bg = "lightblue")
redondeo = tk.BooleanVar()
casilla_redondeo = tk.Checkbutton(cuadro3,
    text="Hacer redondeo",
    bg="lightblue",
    font="consolas 14 bold",
    width=15, anchor=tk.W,
    variable=redondeo)
casilla_redondeo.pack()

simbolo = tk.BooleanVar()
casilla_simbolo = tk.Checkbutton(cuadro3,
    text="Poner símbolo",
    bg="lightblue",
    font="consolas 14 bold",
    width=15, anchor=tk.W,
    variable=simbolo)
casilla_simbolo.pack()

cuadro3.pack(pady=20)

##### FRAME TERCERO #####
cuadro3 = tk.Frame(ventana,
    bg="lightblue")
boton_borrar = tk.Button(cuadro3,
    text="Borrar",
    bg="grey",
    font="consolas, 18 bold",
    width=10,command=borrar)
boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
boton_convertir = tk.Button(cuadro3,
    text="Convertir",
    bg="orange",
    font="consolas, 18 bold",
    width=10, command=convertir,
    state="normal")
boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
cuadro3.pack()

ventana.mainloop()
```

Los cambios más importantes están en la función convertir() y el bloque de código nuevo para los desplegables.

18.- Método bind con teclas

Método bind:

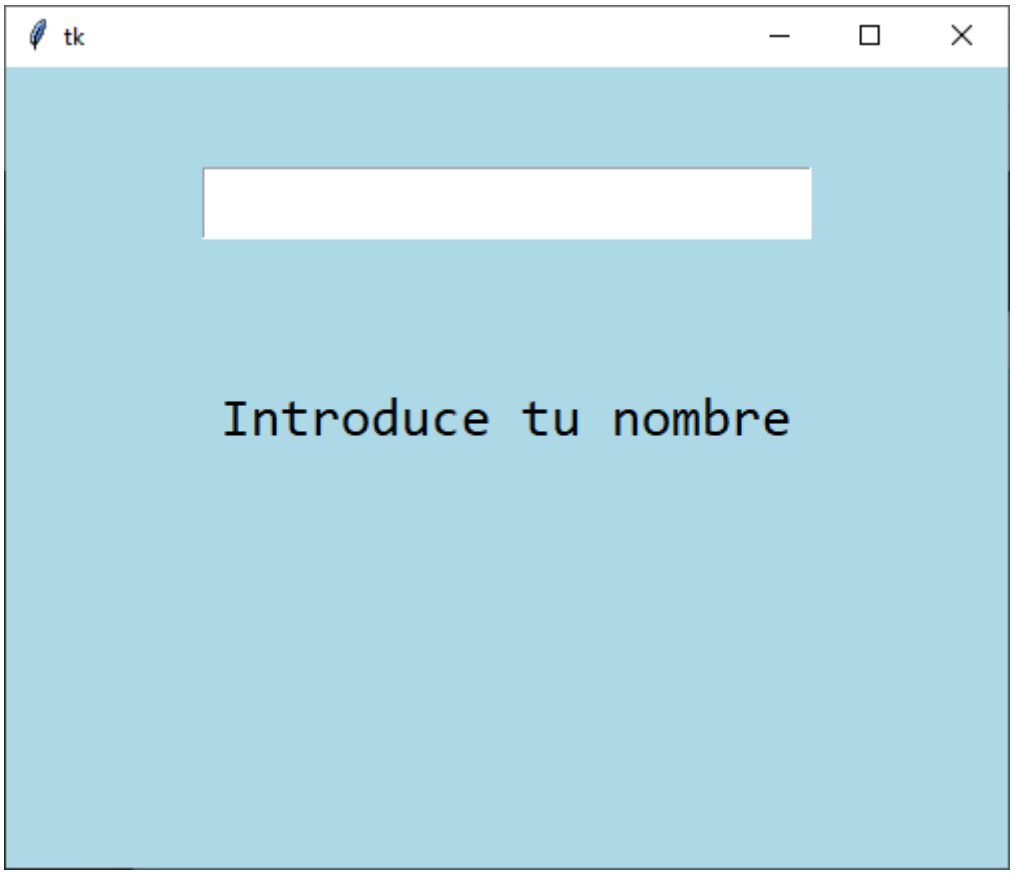
widget.bind(evento, función)

Que nos va a permitir capturar eventos que ocurren en la ventana de tkinter, presionar una tecla, mover el ratón, accionar un widget.

Cuando un widget no dispone de la opción command.

Cuando queremos vincular una acción a un evento directamente (pulsar una tecla, mover el ratón, etc).

```
1   import tkinter as tk
2
3   ###### VENTANA PRINCIPAL ######
4   ventana = tk.Tk()
5   ventana.geometry("500x400+600+100")
6   ventana.configure(bg="lightblue")
7
8   ######## FUNCIONES #########
9
10  ######## widgets #######
11  entrada = tk.Entry(ventana,
12      font="consolas 20")
13  entrada.pack(pady=50)
14
15  rotulo = tk.Label(ventana,
16      text="Introduce tu nombre",
17      font="consolas 20",
18      bg="lightblue")
19  rotulo.pack(pady=20)
20  ##### BUCLE PRINCIPAL #####
21  rotulo.mainloop()
```



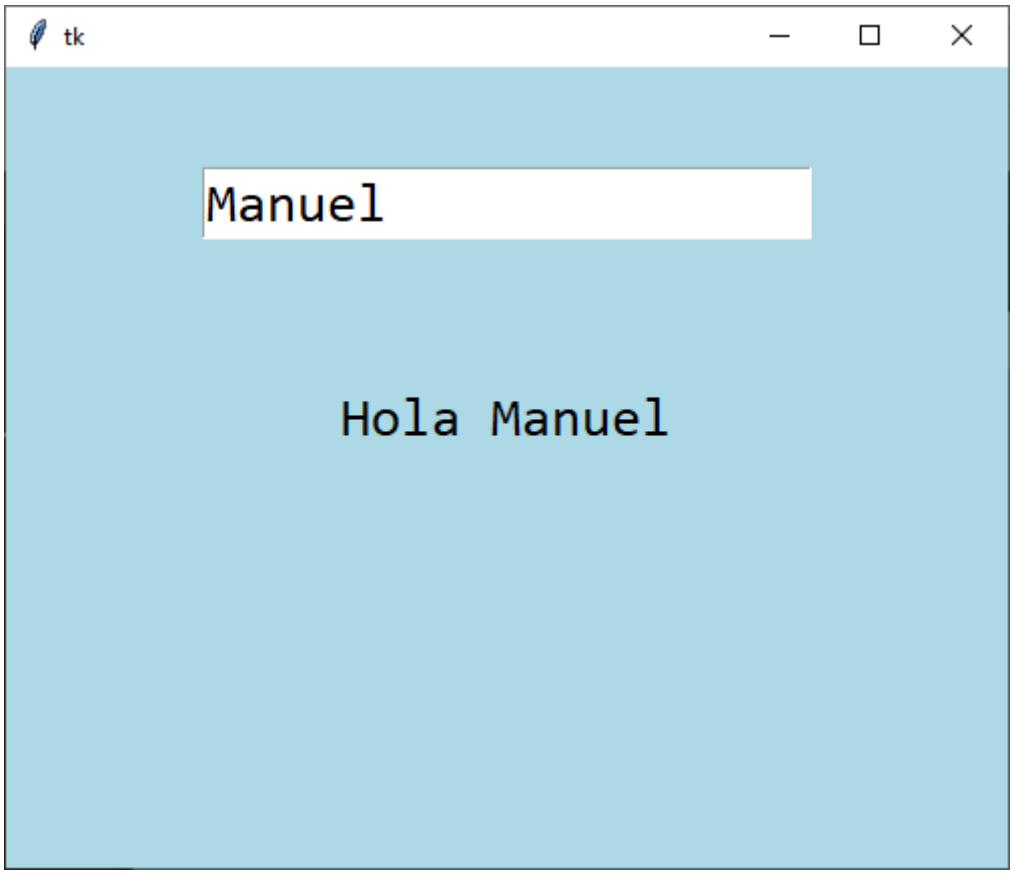
Vamos a poner un nombre y al pulsar Enter que se ejecute la función.

```
13  ####### widgets #######
14  entrada = tk.Entry(ventana,
15      font="consolas 20")
16  entrada.pack(pady=50)
17  entrada.bind("<Return>", saludar)
```

Cuando demos a la tecla Return ejecutará la función saludar().

```
 8  ####### FUNCIONES #########
 9  def saludar(event):
10      nombre = entrada.get()
11      rotulo.config(text=f"Hola {nombre}")
```

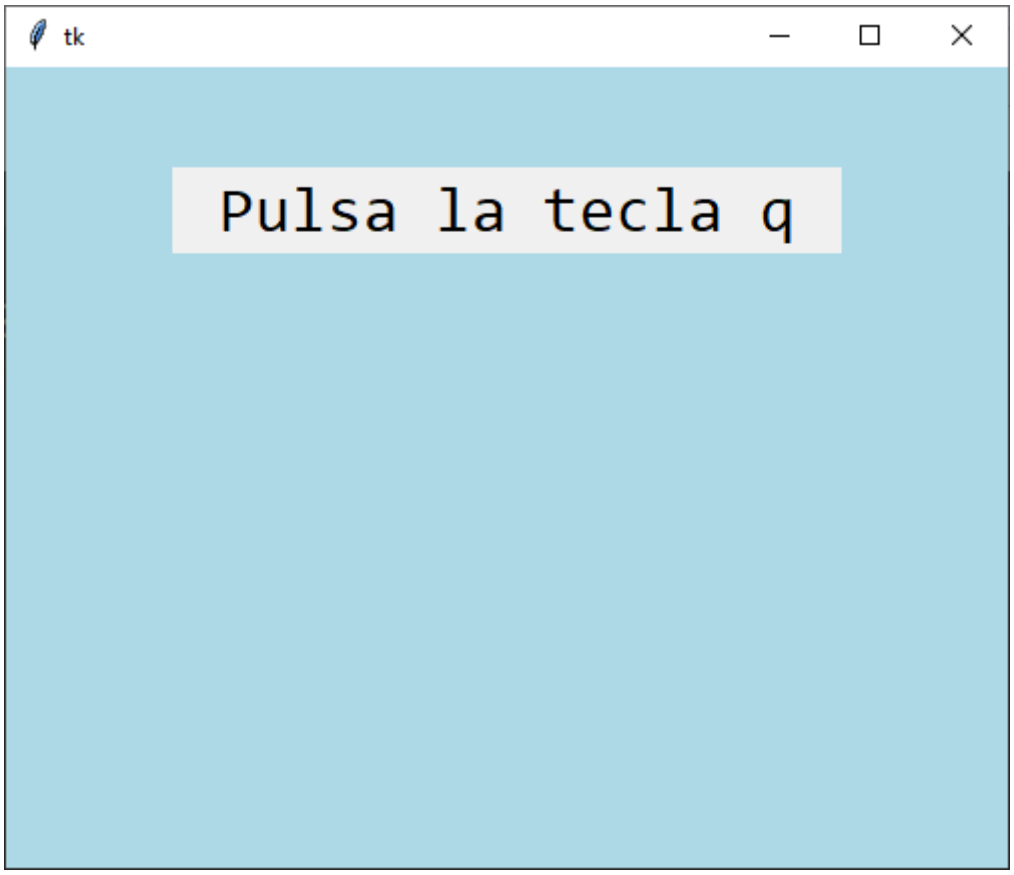
En el rótulo saldrá el mensaje Hola, seguido del nombre que tu hayas introducido en la caja de texto.



Vamos a ver otro ejemplo:

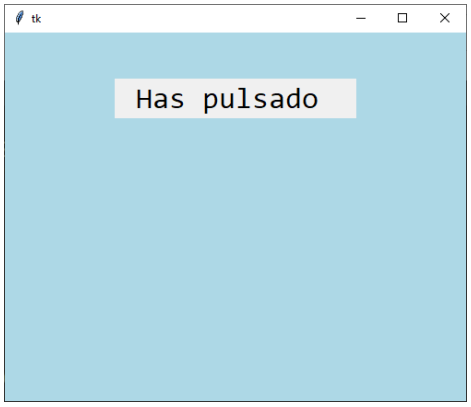
```
1   import tkinter as tk
2
3   ###### VENTANA PRINCIPAL ######
4   ventana = tk.Tk()
5   ventana.geometry("500x400+600+100")
6   ventana.configure(bg="lightblue")
7
8   ######## FUNCIONES #########
9
10
11
12  ######## WIDGETS ########
13  rotulo = tk.Label(ventana,
14      text="Pulsa la tecla q",
15      font="consolas 24")
16  rotulo.pack(pady=50, ipadx=20)
```

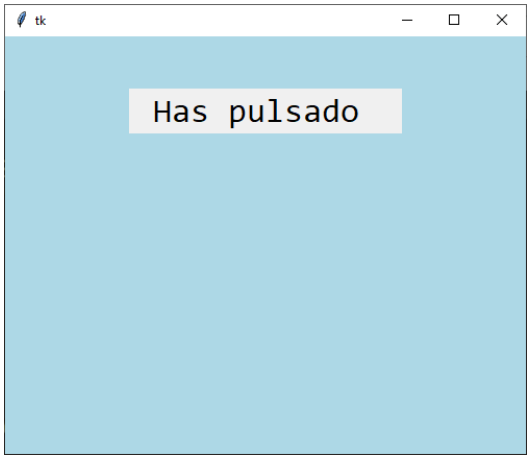
```
17
18  ##### BUCLE PRINCIPAL #####
19  rotulo.mainloop()
```

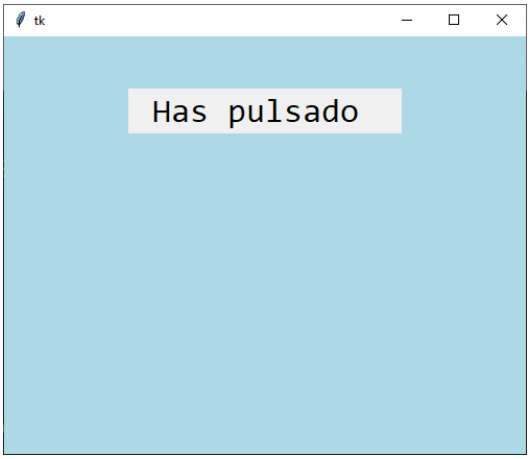



```
1   import tkinter as tk
2   ###### VENTANA PRINCIPAL ######
3   ventana = tk.Tk()
4   ventana.geometry("500x400+600+100")
5   ventana.configure(bg="lightblue")
6
7   ####### FUNCIONES #########
8   def pulsar(event):
9       if event.char == "q":
10          rotulo.config(text=f"Muy bien hecho")
11      else:
12          rotulo.config(text=f"Has pulsado {event.char}")
13
```

```
14  ####### WIDGETS #######
15  rotulo = tk.Label(ventana,
16      text="Pulsa la tecla q",
17      font="consolas 24")
18  rotulo.pack(pady=50, ipadx=20)
19  ventana.bind("<Key>", pulsar)
20  ##### BUCLE PRINCIPAL #####
21  rotulo.mainloop()
```

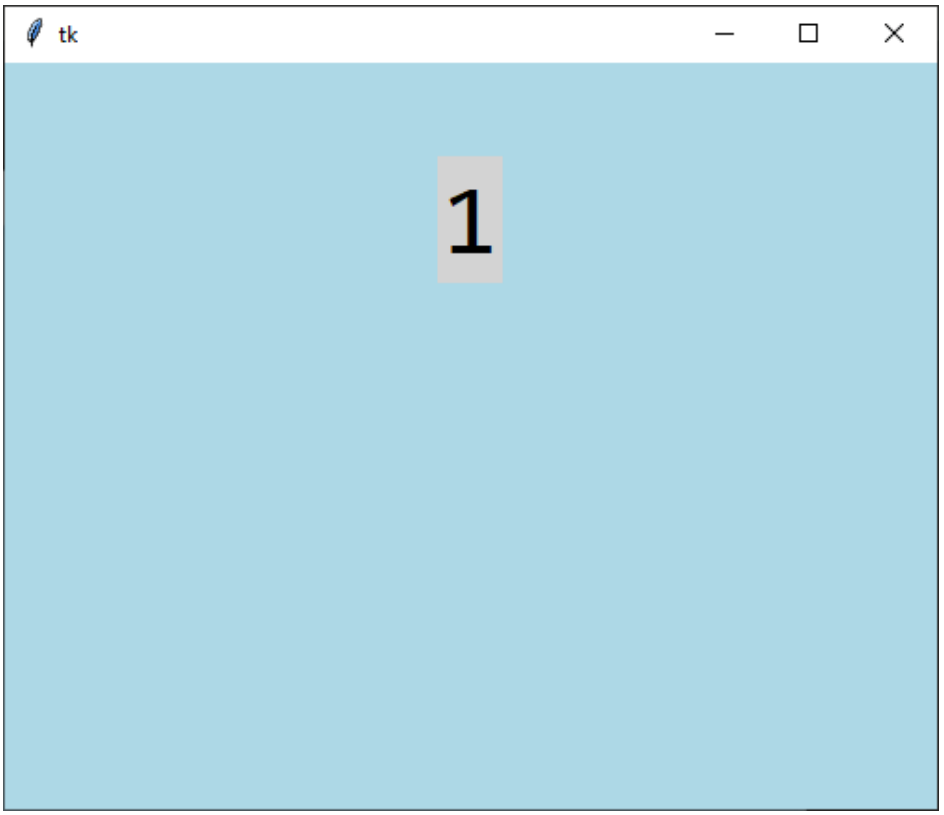




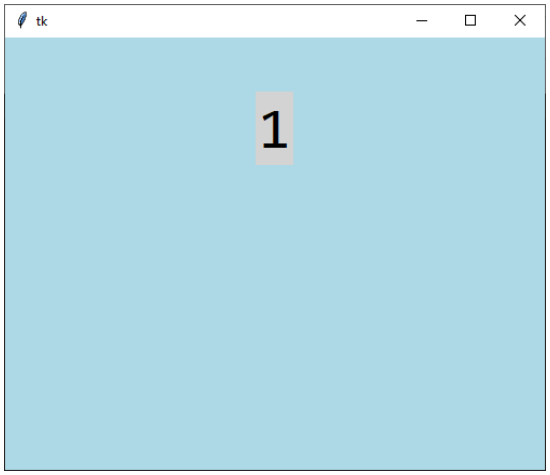


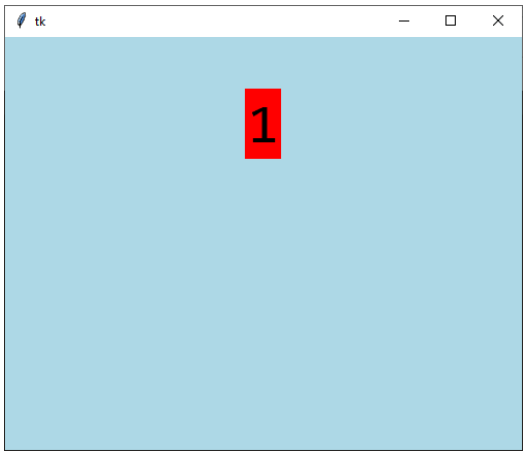
19.- Método bind con eventos del ratón

```
1   import tkinter as tk
2   ###### VENTANA PRINCIPAL ######
3   ventana = tk.Tk()
4   ventana.geometry("500x400+600+100")
5   ventana.configure(bg="lightblue")
6
7   ######## FUNCIONES #########
8
9
10  ######## WIDGETS #######
11  rotulo = tk.Label(ventana,
12      text="1",
13      bg="lightgrey",
14      font="consolas 40")
15  rotulo.pack(pady=50)
16
17  ##### BUCLE PRINCIPAL #####
18  ventana.mainloop()
```

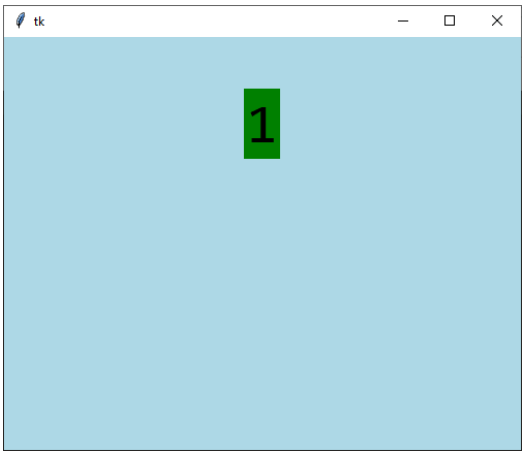


```python
import tkinter as tk
###### VENTANA PRINCIPAL ######
ventana = tk.Tk()
ventana.geometry("500x400+600+100")
ventana.configure(bg="lightblue")

####### FUNCIONES #########
def cambiar_rojo(event):
    rotulo.config(bg="red")

def cambiar_verde(event):
    rotulo.config(bg="green")

####### WIDGETS #######
rotulo = tk.Label(ventana,
    text="1",
    bg="lightgrey",
    font="consolas 40")
rotulo.pack(pady=50)
rotulo.bind("<Button-1>", cambiar_rojo)
rotulo.bind("<Button-3>", cambiar_verde)

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```





Botón izquierdo del ratón



Botón derecho del ratón

```
7   ######## FUNCIONES #########
8   def cambiar_rojo(event):
9       rotulo.config(bg="red")
10      rotulo.unbind("<Button-3>")
11
12  def cambiar_verde(event):
13      rotulo.config(bg="green")
14      rotulo.unbind("<Button-1>")
```

Con la función unbind lo que hacemos es inutilizar el otro botón.

Vamos a crear un botón:

```
30  boton = tk.Button(ventana,
31      text="Reiniciar",
32      bg="lightgrey",
33      font="consolas 20",
34      command=reiniciar)
35  boton.pack(pady=20)
```

Creamos un botón que al hacer clic ejecuta la función reiniciar().

```
7   ######## FUNCIONES #########
8   def cambiar_rojo(event):
9       rotulo.config(bg="red")
10      rotulo.unbind("<Button-3>")
11
```

```
12  def cambiar_verde(event):
13      rotulo.config(bg="green")
14      rotulo.unbind("<Button-1>")
15
16  def reiniciar():
17      rotulo.config(bg="lightgrey")
18      rotulo.bind("<Button-1>", cambiar_rojo)
19      rotulo.bind("<Button-3>", cambiar_verde)
```

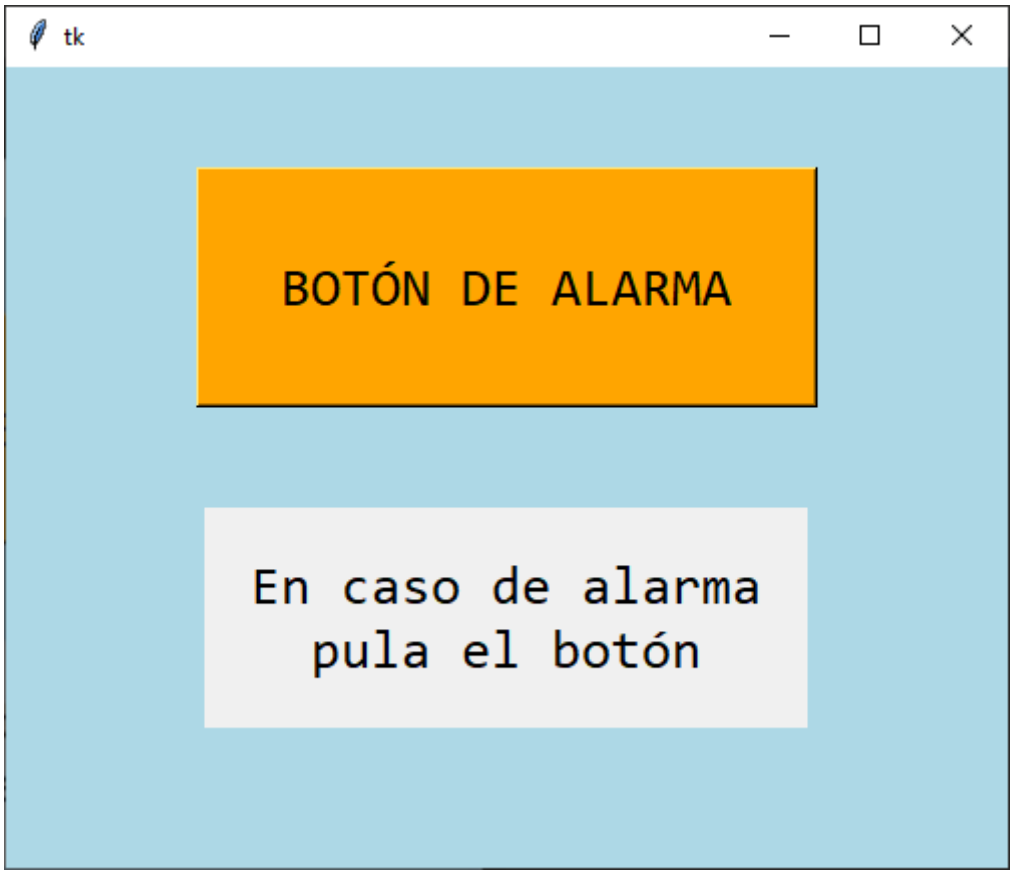
bind ➔ Conectar un evento con una función.

unbind ➔ Desconectar un evento con una función.

Vamos a ver otro ejemplo de como vincular a los eventos generados por el ratón con algún tipo de acción.

```
1   import tkinter as tk
2   ###### VENTANA PRINCIPAL ######
3   ventana = tk.Tk()
4   ventana.geometry("500x400+600+100")
5   ventana.configure(bg="lightblue")
6
7   ####### FUNCIONES #########
8   def pulsar():
9       rotulo.config(text="Pulsado el botón\nde alarma",
10          bg="red3", fg="white", font="consolas 20")
11
12  ######## WIDGETS ########
13
14  boton = tk.Button(ventana,
15      text="BOTÓN DE ALARMA",
16      bg="orange",
17      font="consolas 20",
18      width=20, height=3,
19      command=pulsar)
20  boton.pack(pady=50)
21
22  rotulo = tk.Label(ventana,
23      text="En caso de alarma\npula el botón",
24          font="consolas 20")
```

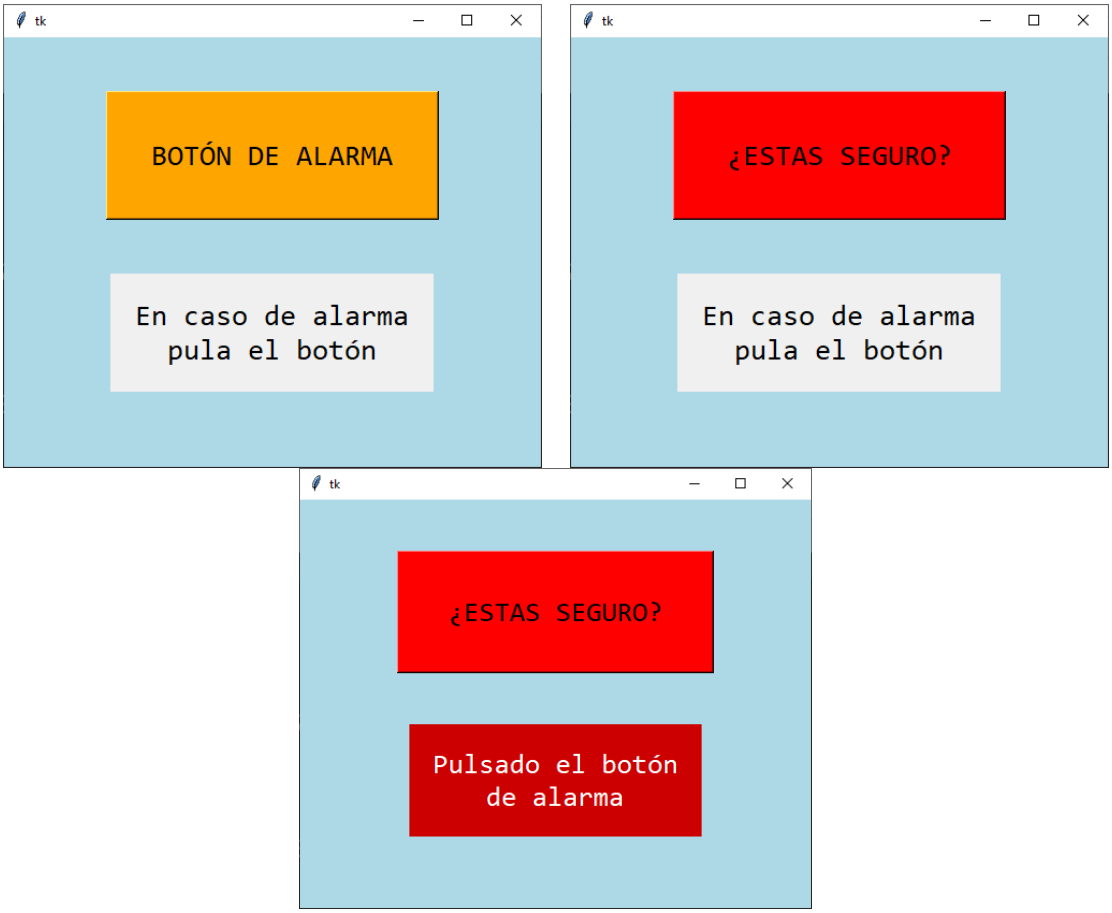
```
24          font="consolas 20")
25      rotulo.pack(ipadx=20, ipady=20)
26
27      ##### BUCLE PRINCIPAL #####
28      ventana.mainloop()
```



Aquí tienes todo el código:

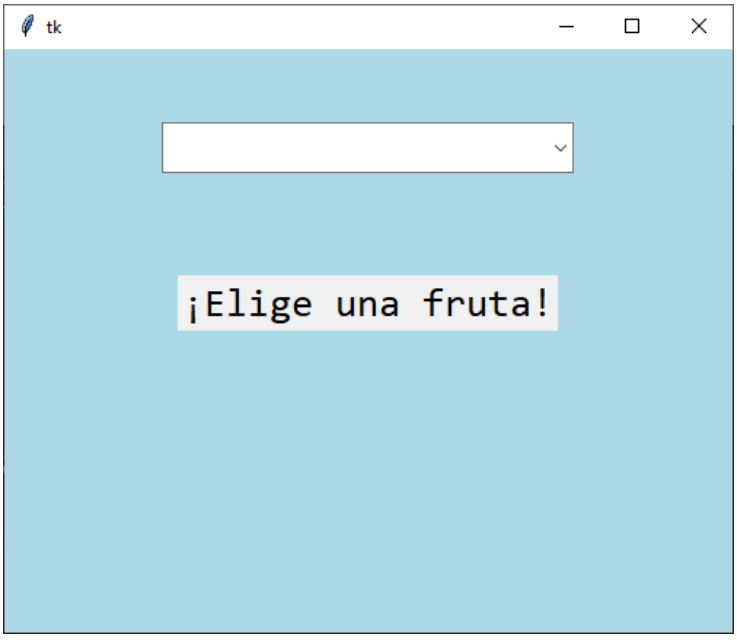
```
import tkinter as tk
###### VENTANA PRINCIPAL ######
ventana = tk.Tk()
ventana.geometry("500x400+600+100")
ventana.configure(bg="lightblue")

####### FUNCIONES #########
def pulsar():
    rotulo.config(text="Pulsado el botón\nde alarma",
        bg="red3", fg="white", font="consolas 20")

def entrar(event):
    boton.config(text="¿ESTAS SEGURO?", bg="red")

def salir(event):
     boton.config(text="BOTÓN DE ALARMA",bg="orange")
```

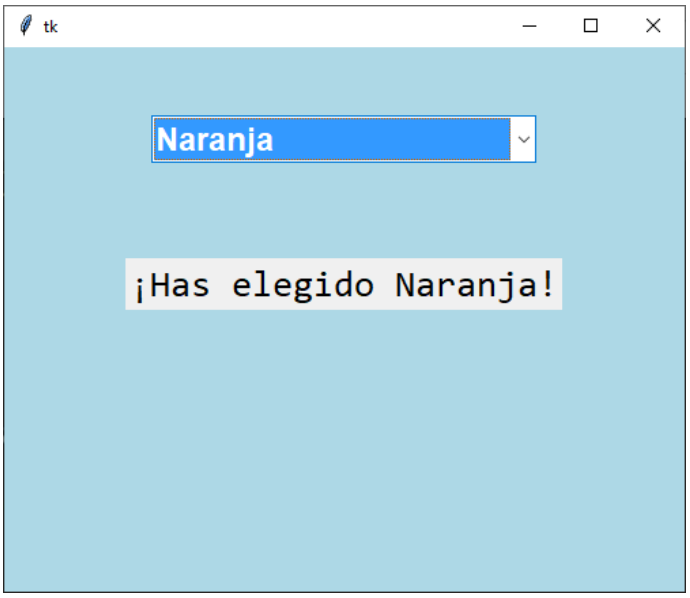
```
######## WIDGETS ########

boton = tk.Button(ventana,
    text="BOTÓN DE ALARMA",
    bg="orange",
    font="consolas 20",
    width=20, height=3,
    command=pulsar)
boton.pack(pady=50)

boton.bind("<Enter>", entrar)
boton.bind("<Leave>", salir)
rotulo = tk.Label(ventana,
    text="En caso de alarma\npula el botón",
        font="consolas 20")
rotulo.pack(ipadx=20, ipady=20)

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```



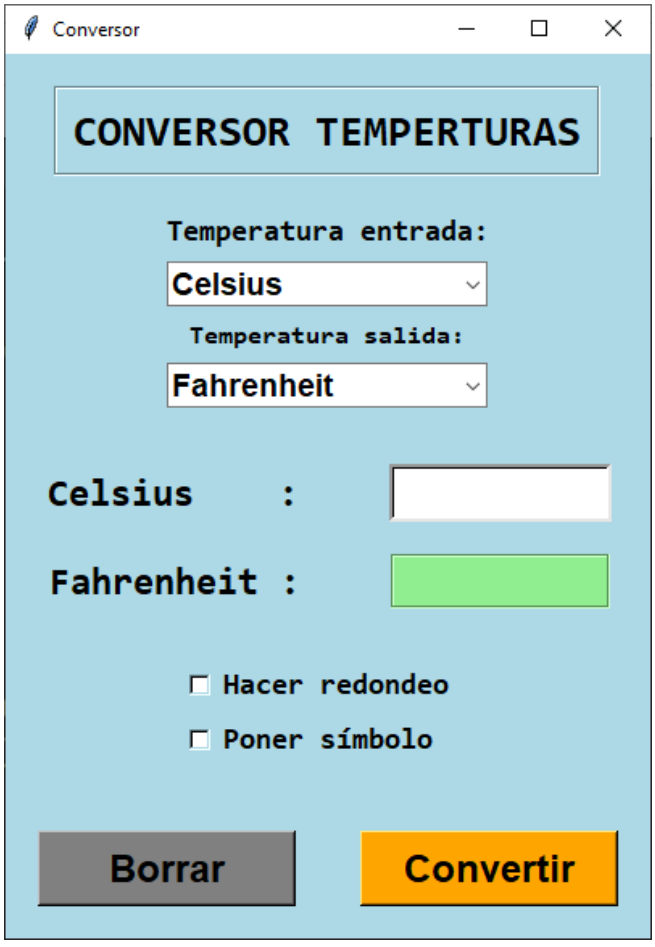
20.- Método bind con el widget Combobox

```
1   import tkinter as tk
2   from tkinter import ttk
3
4   ######## VENTANA PRINCIPAL ########
5   ventana = tk.Tk()
6   ventana.geometry("500x400+600+100")
7   ventana.configure(bg="lightblue")
8
9   ######## FUNCIONES ########
10  def elegir(event):
11      rotulo.config(text=f"¡Has elegido {fruta.get()}!")
12
13  ######## WIDGETS ########
14  fruta = tk.StringVar()
15  desplegable = ttk.Combobox(ventana,
16      font="consolas, 18 bold",
17      values=["Manzana", "Pera", "Melón", "Naranja", "Melocotón"],
18      state="readonly",
19      textvariable=fruta)
20  desplegable.pack(pady=50)
21  desplegable.bind("<<ComboboxSelected>>", elegir)
22
23  rotulo = tk.Label(ventana,
24      text="¡Elige una fruta!",
25      font="consolas 20")
26  rotulo.pack(pady=20)
27  ventana.mainloop()
```





Vamos a volver al proyecto CONVERSOR DE TEMPERATURA:



Este es el código completo:

```
import tkinter as tk
from tkinter import ttk

ventana = tk.Tk()
```

```
ventana.title("Conversor")
ventana.geometry("400x500+700+100")
ventana.minsize(width=400, height=550)
ventana.configure(bg="lightblue")

######## FUNCIONES ########
def convertir():
    try:
        entrada = temp_entrada.get()
        salida = temp_salida.get()
        grados = float(entrada_grados.get())
        if entrada == salida:
            resultado = grados
        elif entrada == "Celsius" and salida == "Fahrenheit":
            resultado = round(grados * 1.8 +32, 2)
        elif entrada == "Fahrenheit" and salida == "Celsius":
            resultado = round((grados - 32)/ 1.8, 2)
        elif entrada == "Celsius" and salida == "Kelvin":
            resultado = round(grados + 273.15, 2)
        elif entrada == "Kelvin" and salida == "Celsius":
            resultado = round(grados - 273.15, 2)
        elif entrada == "fahrenheit" and salida == "Kelvin":
            resultado = round(((grados - 32)/1.8)+273.15, 2)

        if redondeo.get():
            resultado = round(resultado)
        if simbolo.get():
            if salida == "Celsius":
                resultado = f"{resultado}\u00B0C"
            elif salida == "Fahrenheit":
                resultado = f"{resultado}\u00B0F"
            else:
                resultado = f"{resultado} K"
        rotulo_resultado.config(text=resultado)
        entrada_grados.config(state="disabled")
        boton_convertir.config(state="disabled")
    except ValueError:
        if grados == "":
            rotulo_resultado.config(text="¿Dato?")
        else:
            rotulo_resultado.config(text="Error")
    finally:
        entrada_grados.config(state="disabled")
        boton_convertir.config(state="disabled")

def mostrar_entrada(event):
    opcion_entrada = temp_entrada.get()
    opcion_salida = temp_salida.get()
```

```
    if opcion_entrada == "Celsius":
        rotulo_primero.config(text="Celsius      :")
        if opcion_salida == "Celsius":
            rotulo_segundo.config(text="Fahrenheit  :")
            desplegable_salida.set("Fahrenheit")
    elif opcion_entrada == "Fahrenheit":
        rotulo_primero.config(text="Fahrenheit  :")
        if opcion_salida == "Fahrenheit":
            rotulo_segundo.config(text="Celsius     :")
            desplegable_salida.set("Celsius")
    elif opcion_entrada == "Kelvin":
        rotulo_primero.config(text="Kelvin      :")
        if opcion_salida == "Kelvin":
            rotulo_segundo.config(text="Celsius   :")
            desplegable_salida.set("Celsius")

    entrada_grados.focus()
```

```
def mostrar_salida(event):
    opcion_entrada = temp_entrada.get()
    opcion_salida = temp_salida.get()

    if opcion_salida == "Celsius":
        rotulo_segundo.config(text="Celsius      :")
        if opcion_entrada == "Celsius":
            rotulo_primero.config(text="Fahrenheit  :")
            desplegable_entrada.set("Fahrenheit")
    elif opcion_salida == "Fahrenheit":
        rotulo_segundo.config(text="Fahrenheit  :")
        if opcion_entrada == "Fahrenheit":
            rotulo_primero.config(text="Celsius       :")
            desplegable_entrada.set("Celsius")
    elif opcion_salida == "Kelvin":
        rotulo_segundo.config(text="Kelvin      :")
        if opcion_entrada == "Kelvin":
            rotulo_primero.config(text="Fahrenheit :")
            desplegable_entrada.set("Fahrenheit")
```

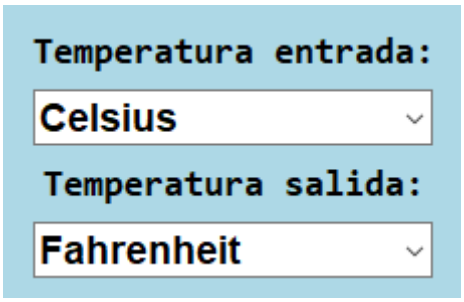
```
def borrar():
    entrada_grados.config(state="normal")
    boton_convertir.config(state="normal")
    entrada_grados.delete(0, tk.END)
    rotulo_resultado.config(text="")

##### LABEL TITULO #####
rotulo_titulo = tk.Label(ventana,
    text="CONVERSOR TEMPERTURAS",
    bg="lightblue", fg="black",
    font= "consolas 20 bold",
```

```
    relief=tk.GROOVE, bd=2,
    padx=10, pady=10)
rotulo_titulo.pack(padx=20, pady=20)

###### DESPLEGABLES CON SUS RÓTULOS ######
rotulo_entrada = tk.Label(ventana,
    text="Temperatura entrada:",
    bg="lightblue", fg="black",
    font="consolas 14 bold",
    bd=2)
rotulo_entrada.pack()

temp_entrada = tk.StringVar()
desplegable_entrada = ttk.Combobox(ventana,
    font="consola 14 bold",
    width=16,
    values=["Celsius", "Fahrenheit", "Kelvin"],
    state="readonly",
    textvariable=temp_entrada)
desplegable_entrada.pack(pady=5)
desplegable_entrada.set("Celsius")

rotulo_salida = tk.Label(ventana,
    text="Temperatura salida:",
    bg="lightblue", fg="black",
    font="consolas 14 bold",
    bd=2)
rotulo_salida.pack()
temp_salida = tk.StringVar()
desplegable_salida = ttk.Combobox(ventana,
    font="consola 14 bold",
    width=16,
    values=["Celsius", "Fahrenheit", "Kelvin"],
    state="readonly",
    textvariable=temp_salida)
desplegable_salida.pack(pady=5)
desplegable_salida.set("Fahrenheit")

desplegable_entrada.bind("<<ComboboxSelected>>", mostrar_entrada)
desplegable_salida.bind("<<ComboboxSelected>>", mostrar_salida)

##### CUADRO ESPACIO #####
cuadro_espacio = tk.Frame(ventana,
    height=20,
    bg="lightblue")
cuadro_espacio.pack()

##### FRAME PRIMERO #####
cuadro1 = tk.Frame(ventana,
```

```
    bg="lightblue")
rotulo_primero = tk.Label(cuadro1,
    text="Celsius    : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_primero.pack(side=tk.LEFT, padx=20, pady=0)
entrada_grados= tk.Entry(cuadro1,
    bg="white",fg="black",
    font="consolas 18 bold",
    relief=tk.SUNKEN, bd=3,
    width=10,
    justify=tk.RIGHT,
    state="normal")
entrada_grados.pack(side=tk.LEFT, padx=20, pady=0)
cuadro1.pack(pady=10)

##### FRAME SEGUNDO #####
cuadro2 = tk.Frame(ventana,
    bg="lightblue")
rotulo_segundo = tk.Label(cuadro2,
    text="Fahrenheit : ",
    bg="lightblue",
    font="consolas 18 bold")
rotulo_segundo.pack(side=tk.LEFT, padx=20, pady=0)
rotulo_resultado = tk.Label(cuadro2,
    text="",
    bg="lightgreen",
    relief="groove",
    font="consolas 18 bold",
    width=10,
    anchor=tk.E)
rotulo_resultado.pack(side=tk.LEFT, padx=20, pady=0)
cuadro2.pack(pady=10)

##### CUADRO TERCERO #####
cuadro3 = tk.Frame(ventana,
bg = "lightblue")
redondeo = tk.BooleanVar()
casilla_redondeo = tk.Checkbutton(cuadro3,
    text="Hacer redondeo",
    bg="lightblue",
    font="consolas 14 bold",
    width=15, anchor=tk.W,
    variable=redondeo)
casilla_redondeo.pack()

simbolo = tk.BooleanVar()
casilla_simbolo = tk.Checkbutton(cuadro3,
    text="Poner símbolo",
```

```
    bg="lightblue",
    font="consolas 14 bold",
    width=15, anchor=tk.W,
    variable=simbolo)
casilla_simbolo.pack()

cuadro3.pack(pady=20)

##### FRAME TERCERO #####
cuadro3 = tk.Frame(ventana,
    bg="lightblue")
boton_borrar = tk.Button(cuadro3,
    text="Borrar",
    bg="grey",
    font="consolas, 18 bold",
    width=10,command=borrar)
boton_borrar.pack(side=tk.LEFT, padx=20, pady=20)
boton_convertir = tk.Button(cuadro3,
    text="Convertir",
    bg="orange",
    font="consolas, 18 bold",
    width=10, command=convertir,
    state="normal")
boton_convertir.pack(side=tk.LEFT, padx=20, pady=20)
cuadro3.pack()

ventana.mainloop()
```

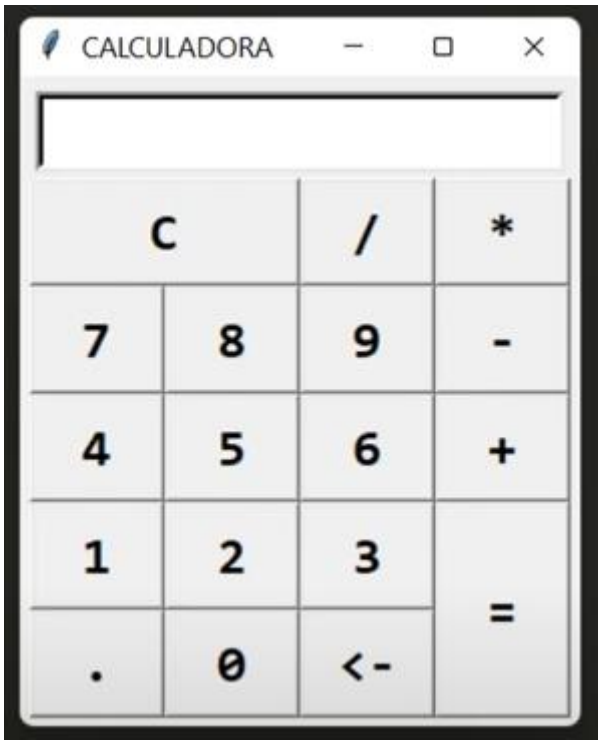
Ahora no podrá coincidir temperatura entra y temperatura salida.



21.- El método grid de tkinter

Hasta ahora los proyectos que hemos realizados los widgets estaban organizados de arriba hacia abajo y de izquierda a derecha.

Pero que pasaría quisiéramos hacer una aplicación con una organización un poco más compleja, como por ejemplo una calculadora.



Que no guardan posiciones simétricas con los demás, por ejemplo el botón de la tecla C ocupa más espacio a lo acho que los demás botones o el botón de la tecla = (igual) ocupa más espacio a lo alto de los demás botones.

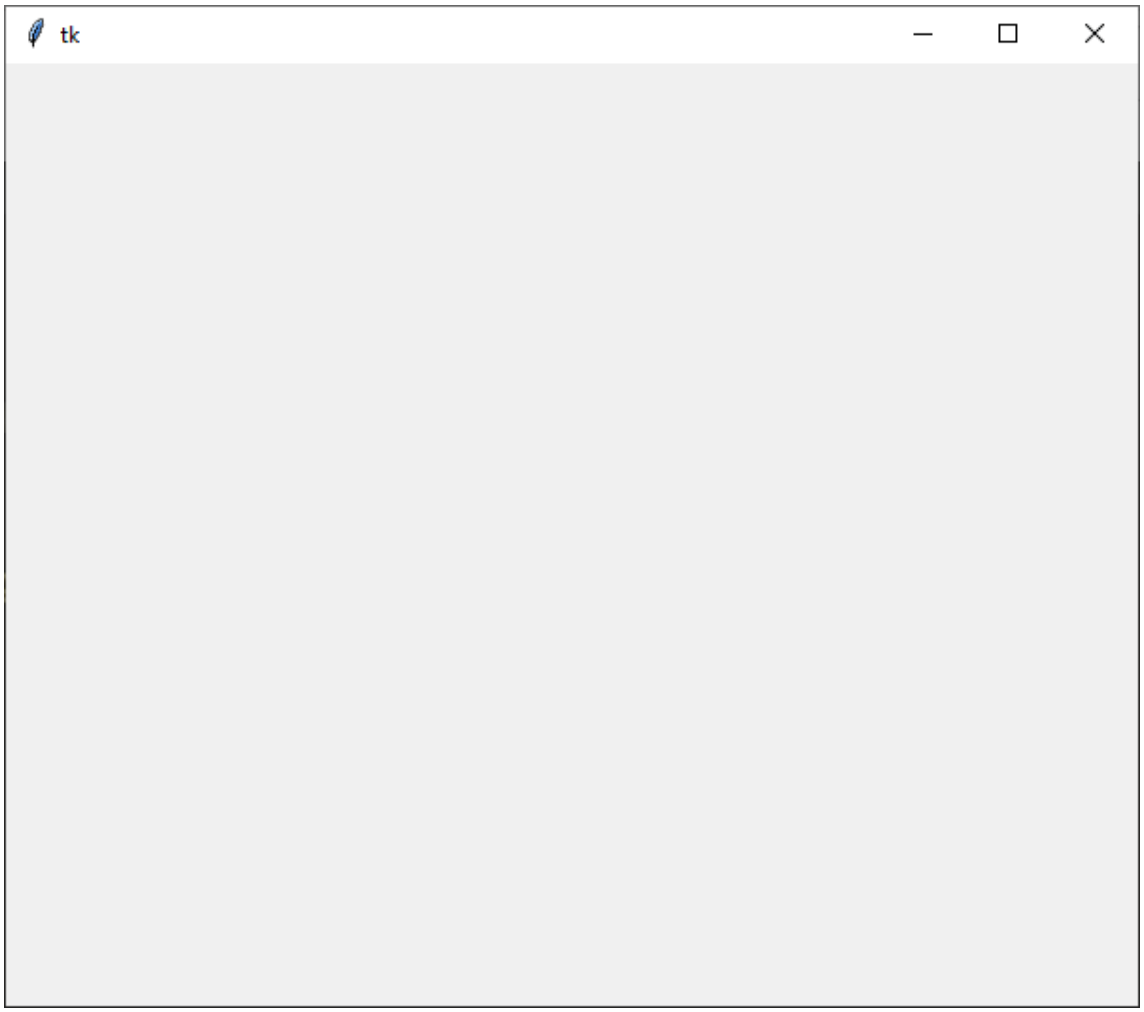
Vamos a disponer de otro método que va a ser el método grid que nos va a permitir llevar a cavo la organización de los widgets en la ventana de una forma más sencilla y rápida.

Vamos a empezar con un nuevo proyecto.

```
1   import tkinter as tk
2
3   ##### VENTANA PRINCIPAL #####
4   ventana = tk.Tk()
5   ventana.geometry("600x500+500+100")
6
7   ##### WIDGETS APLICACIÓN #####
8
9
10  ##### BUCLE PRINCIPAL #####
```

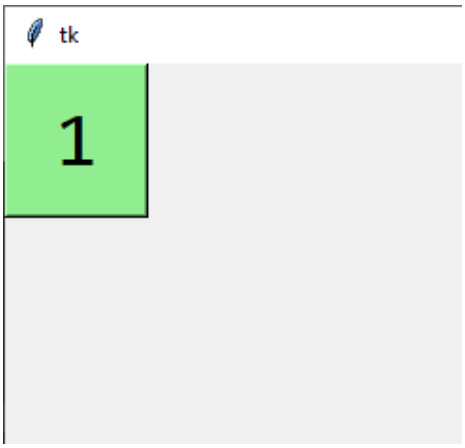
```
11    ventana.mainloop()
```

Partiendo del siguiente proyecto:



Vamos empezar a insertar widgets.

```
7    ##### WIDGETS APLICACIÓN #####
8    uno = tk.Button(ventana,
9        text="1",
10       font="consolas 30",
11       bg="lightgreen",
12       width=3)
13   uno.grid()
```



En este ejemplo lo colocará:

En la parte superior izquierda.

Pero este método permite organizar los widgets en

Filas y columnas.

```
7    ##### WIDGETS APLICACIÓN #####
8    uno = tk.Button(ventana,
9        text="1",
10       font="consolas 30",
11       bg="lightgreen",
12       width=3)
13   uno.grid(row=0, column=0)
```

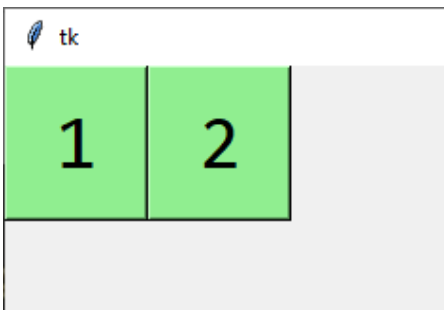
En este caso el resultado será el mismo.

```
uno.grid(row=3, column=7)
```

El resultado va ser el mismo porque este método trabaja con respecto a otros widget que ya están posicionados en la ventana.

```
7    ##### WIDGETS APLICACIÓN #####
8    uno = tk.Button(ventana,
9        text="1",
10       font="consolas 30",
11       bg="lightgreen",
12       width=3)
13   uno.grid(row=0, column=0)
14
15   dos = tk.Button(ventana,
16       text="2",
17       font="consolas 30",
18       bg="lightgreen",
19       width=3)
20   dos.grid(row=0, column=1)
```

Hemos creado un segundo widget en la fila o columna 1, con respecto al anterior widget que está en la fila 0 columna 0.

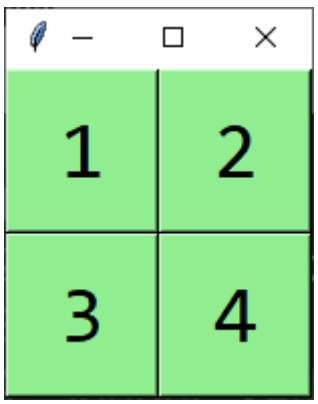


```
22    tres = tk.Button(ventana,
23        text="3",
24        font="consolas 30",
25        bg="lightgreen",
26        width=3)
27    tres.grid(row=1, column=0)
28
29    cuatro = tk.Button(ventana,
30        text="4",
31        font="consolas 30",
32        bg="lightgreen",
33        width=3)
34    cuatro.grid(row=1, column=1)
```

Hemos agregado dos widgets más uno en la fila1 columna 0 y el otro en la fila 1 columna 1.

Vamos a comentar las dimensiones:

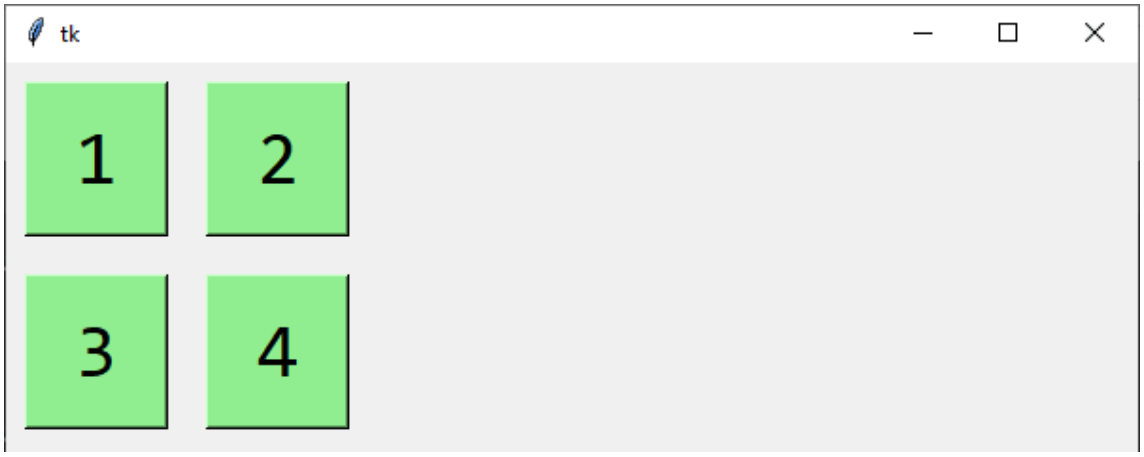
```
5    # ventana.geometry("600x500+500+100")
```



Observamos que la ventana se ajusta a los widgets.

```
7    ##### WIDGETS APLICACIÓN #####
8    uno = tk.Button(ventana,
9        text="1",
10       font="consolas 30",
11       bg="lightgreen",
12       width=3)
13   uno.grid(row=0, column=0, padx=10, pady=10)
```

A todos los widget en el método grid le agregamos el padx y el pady que ya hemos utilizado en capítulos anteriores.

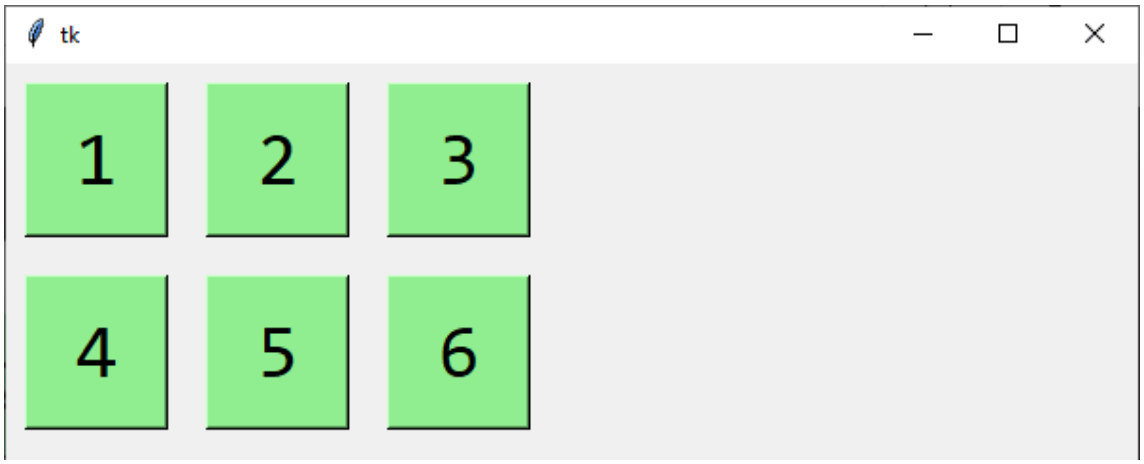


Podemos hacer que un widget ocupe más de una fila o más de una columna.

Vamos añadir dos botones más y modificar sus posiciones:

```
##### WIDGETS APLICACIÓN #####
uno = tk.Button(ventana,
    text="1",
    font="consolas 30",
    bg="lightgreen",
    width=3)
uno.grid(row=0, column=0, padx=10, pady=10)

dos = tk.Button(ventana,
    text="2",
    font="consolas 30",
    bg="lightgreen",
    width=3)
dos.grid(row=0, column=1, padx=10, pady=10)

tres = tk.Button(ventana,
    text="3",
    font="consolas 30",
    bg="lightgreen",
    width=3)
```

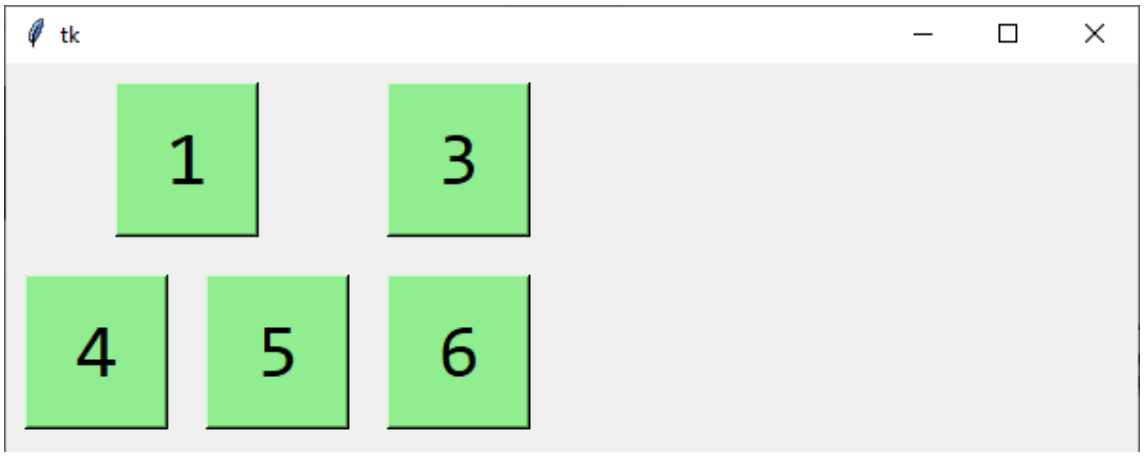
```
tres.grid(row=0, column=2, padx=10, pady=10)

cuatro = tk.Button(ventana,
    text="4",
    font="consolas 30",
    bg="lightgreen",
    width=3)
cuatro.grid(row=1, column=0, padx=10, pady=10)

cinco = tk.Button(ventana,
    text="5",
    font="consolas 30",
    bg="lightgreen",
    width=3)
cinco.grid(row=1, column=1, padx=10, pady=10)

seis = tk.Button(ventana,
    text="6",
    font="consolas 30",
    bg="lightgreen",
    width=3)
seis.grid(row=1, column=2, padx=10, pady=10)
```



Ahora queremos que un widget ocupe los botones 1 y 2.

Eliminaremos el botón 2 para hacer más grande el botón 1.

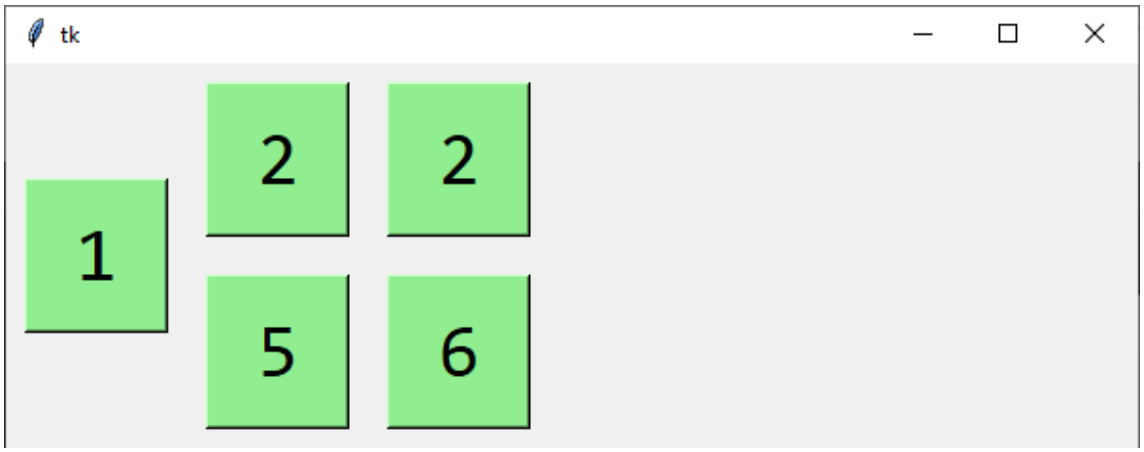
```
 8  uno = tk.Button(ventana,
 9      text="1",
10      font="consolas 30",
11      bg="lightgreen",
12      width=3)
13  uno.grid(row=0, column=0, columnspan=2, padx=10, pady=10)
```



Si queremos que ocupe dos filas.

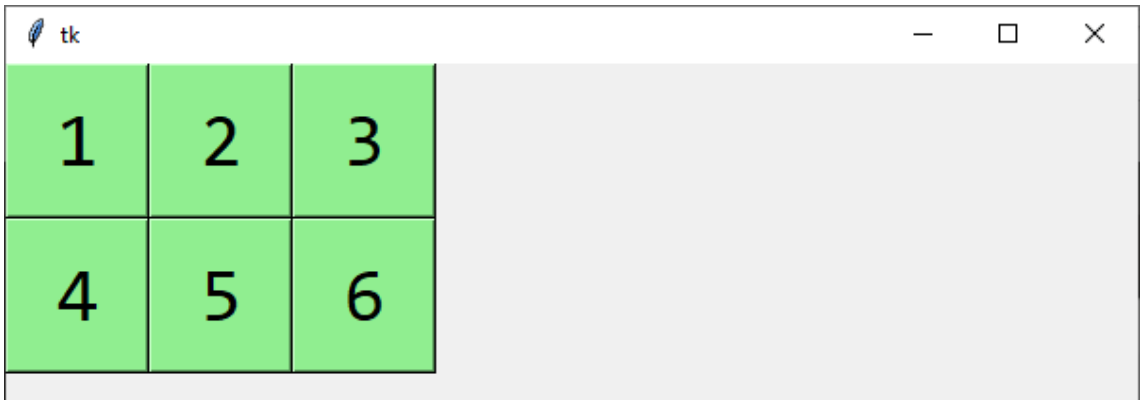
Ahora si borramos el botón 4 y modificamos el botón1.

```
 8  uno = tk.Button(ventana,
 9      text="1",
10      font="consolas 30",
11      bg="lightgreen",
12      width=3)
13  uno.grid(row=0, column=0, rowspan=2, padx=10, pady=10)
```



22.- Opción sticky para el método grid

Vamos a seguir con la aplicación del video anterior.



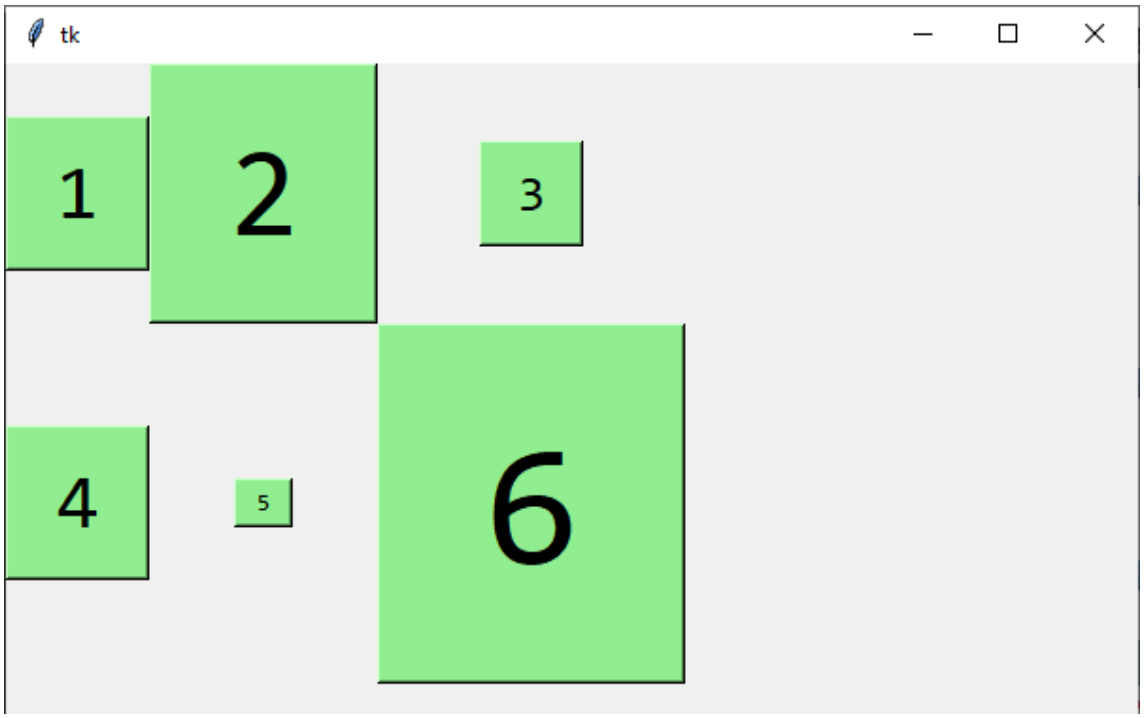
Hemos quitado los pading.

Vamos a cambiar el tamaño de los botones.

Realiza las siguientes modificaciones:

```
##### WIDGETS APLICACIÓN #####
uno = tk.Button(ventana,
    text="1",
    font="consolas 30",
    bg="lightgreen",
    width=3)
uno.grid(row=0, column=0)

dos = tk.Button(ventana,
    text="2",
    font="consolas 50",
    bg="lightgreen",
    width=3)
dos.grid(row=0, column=1)

tres = tk.Button(ventana,
    text="3",
    font="consolas 20",
    bg="lightgreen",
    width=3)
tres.grid(row=0, column=2)

cuatro = tk.Button(ventana,
    text="  4  ",
    font="consolas 30",
    bg="lightgreen",
    width=3)
cuatro.grid(row=1, column=0)

cinco = tk.Button(ventana,
```

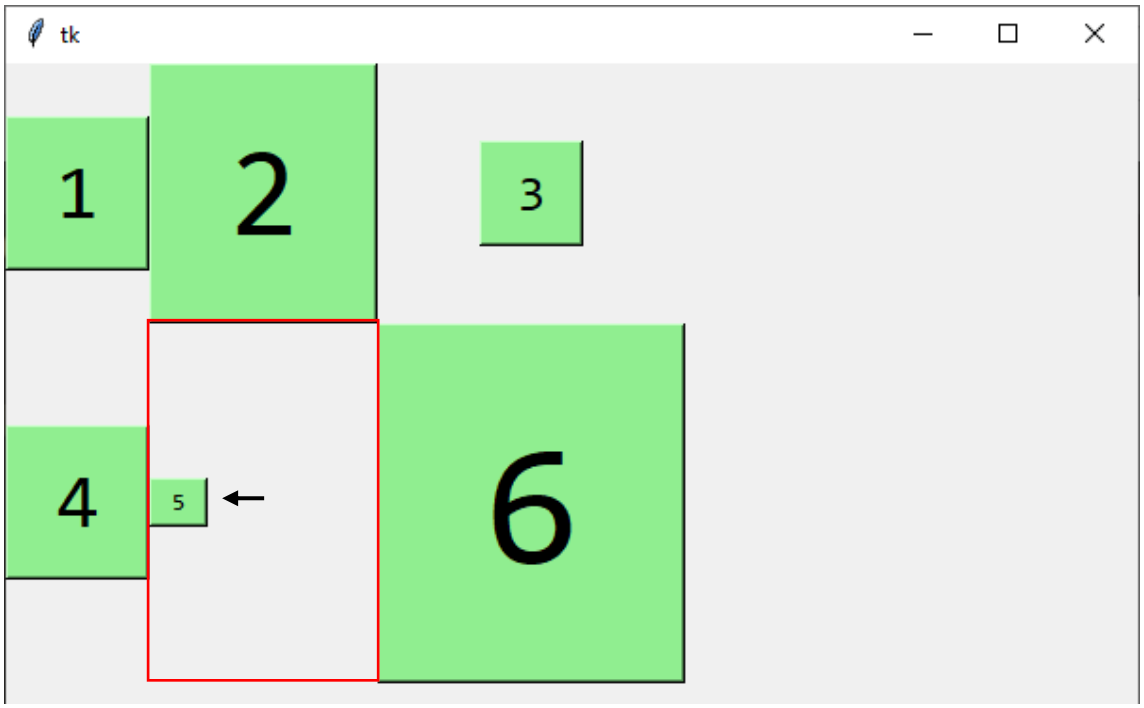
```
    text="5",
    font="consolas 10",
    bg="lightgreen",
    width=3)
cinco.grid(row=1, column=1)

seis = tk.Button(ventana,
    text="6",
    font="consolas 70",
    bg="lightgreen",
    width=3)
seis.grid(row=1, column=2)
```



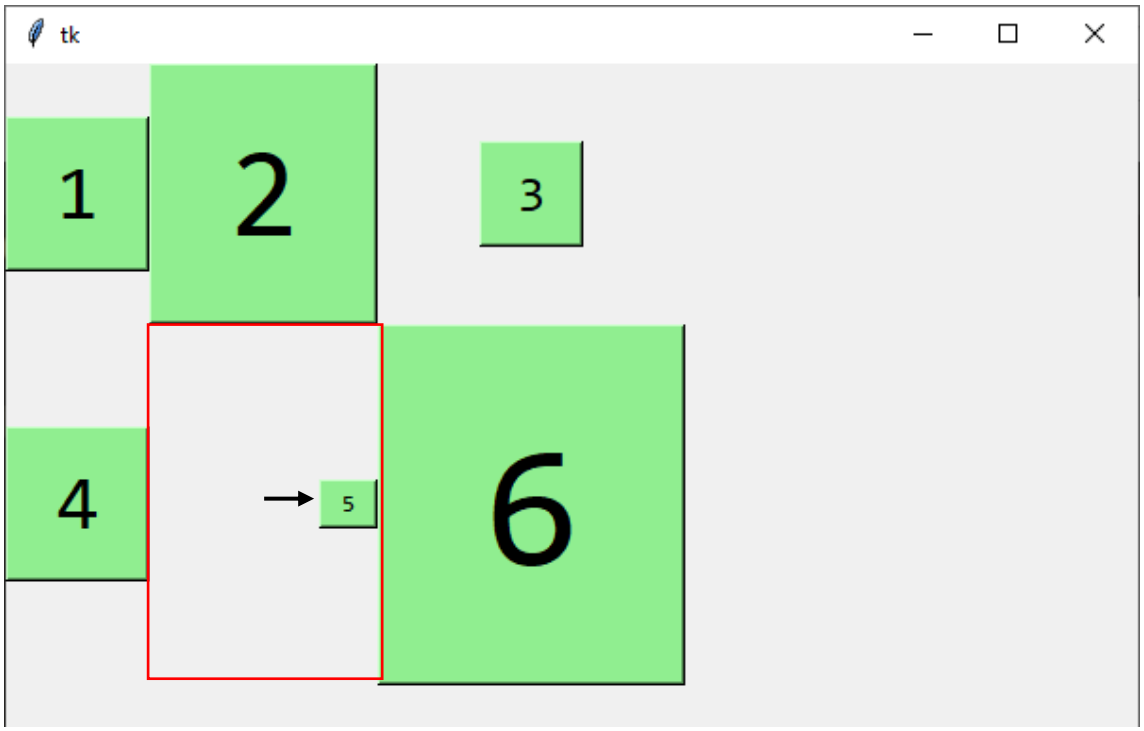
Todos los widgets se ajustan al widget con mayor altura y al widgt con mayor anchura.

Además los widget se colocan en el centro que está reservado para ellos.

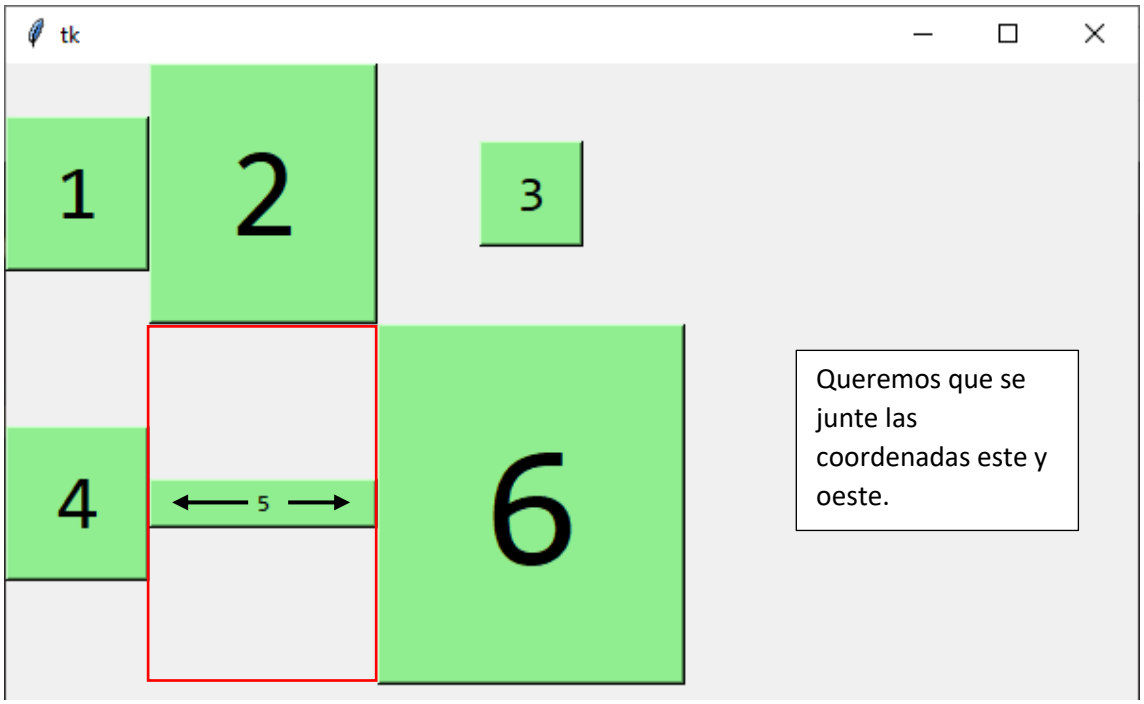
```
36  cinco = tk.Button(ventana,
37      text="5",
38      font="consolas 10",
39      bg="lightgreen",
40      width=3)
41  cinco.grid(row=1, column=1, sticky=tk.W)
```



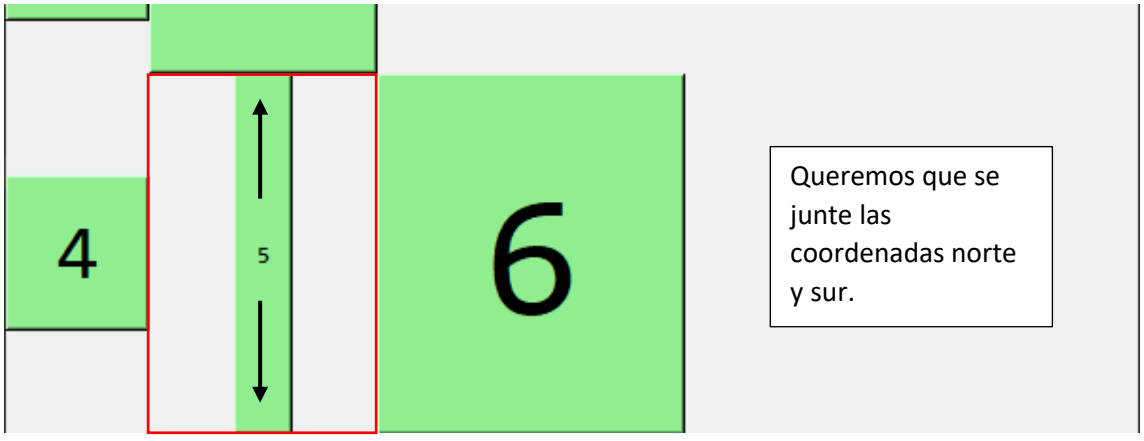

```
36    cinco = tk.Button(ventana,
37        text="5",
38        font="consolas 10",
39        bg="lightgreen",
40        width=3)
41    cinco.grid(row=1, column=1, sticky=tk.E)
```



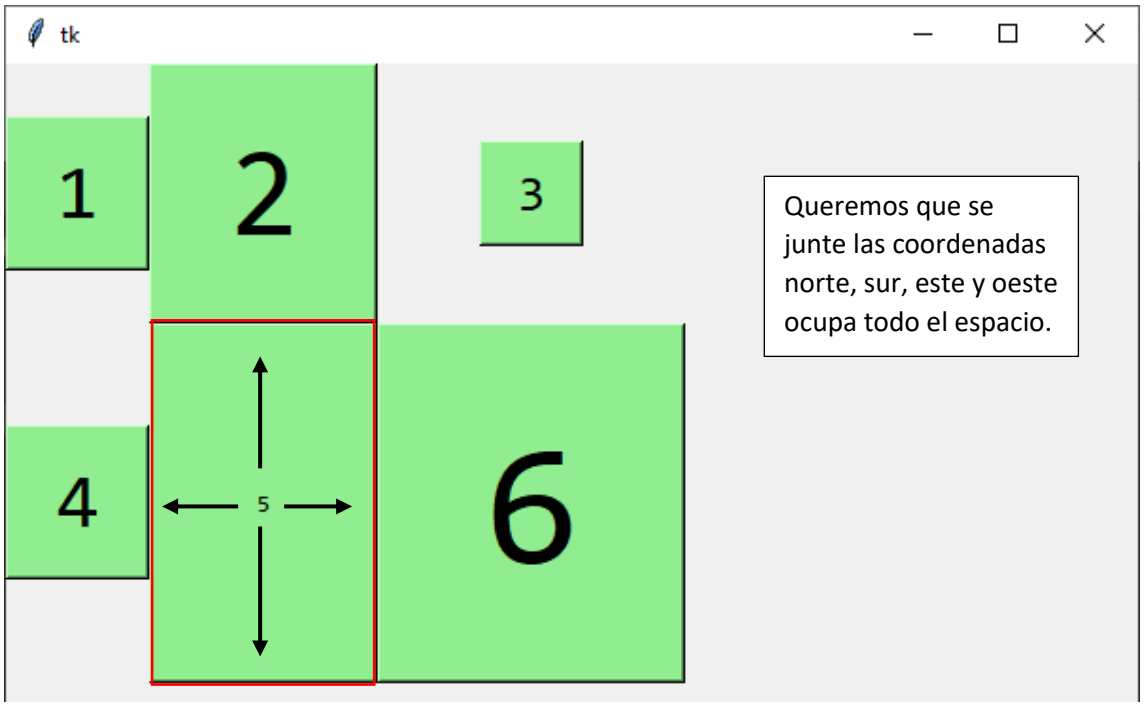
```
36    cinco = tk.Button(ventana,
37        text="5",
38        font="consolas 10",
39        bg="lightgreen",
40        width=3)
41    cinco.grid(row=1, column=1, sticky="ew")
```



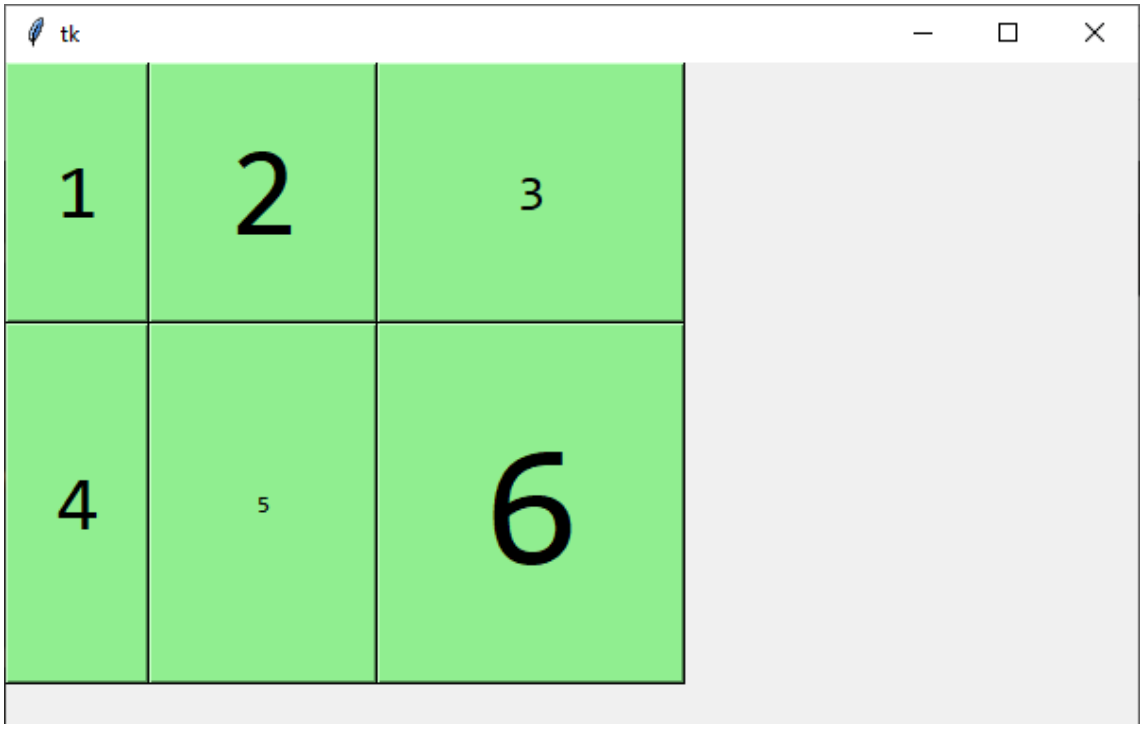

```
36  cinco = tk.Button(ventana,
37      text="5",
38      font="consolas 10",
39      bg="lightgreen",
40      width=3)
41  cinco.grid(row=1, column=1, sticky="ns")
```



```
36    cinco = tk.Button(ventana,
37        text="5",
38        font="consolas 10",
39        bg="lightgreen",
40        width=3)
41    cinco.grid(row=1, column=1, sticky="nsew")
```

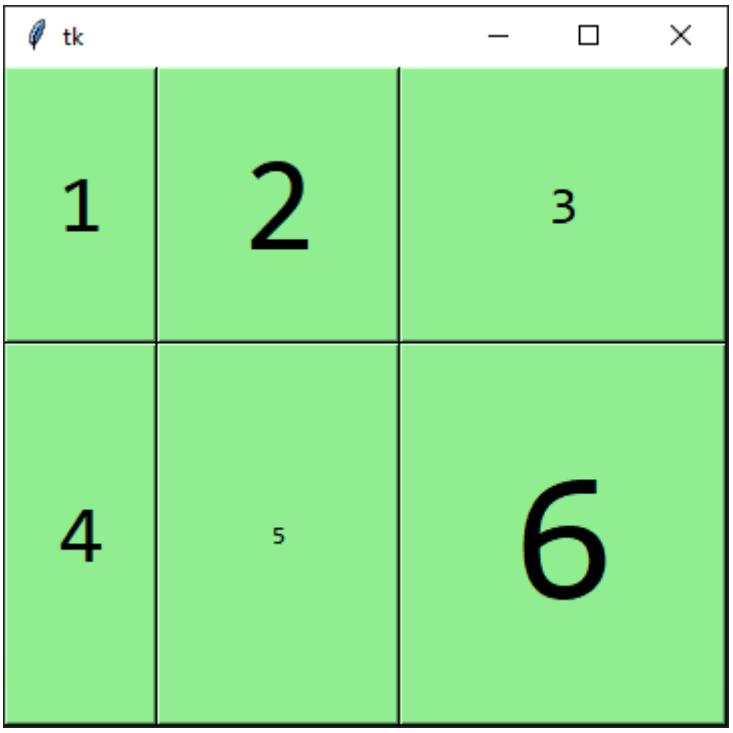


Vamos a ponerlo en todos los widgets.



```
5    # ventana.geometry("600x500+500+100")
```

Si comentamos las dimensiones predeterminadas.

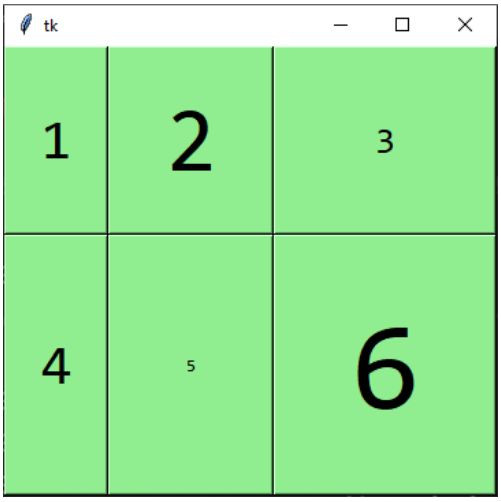


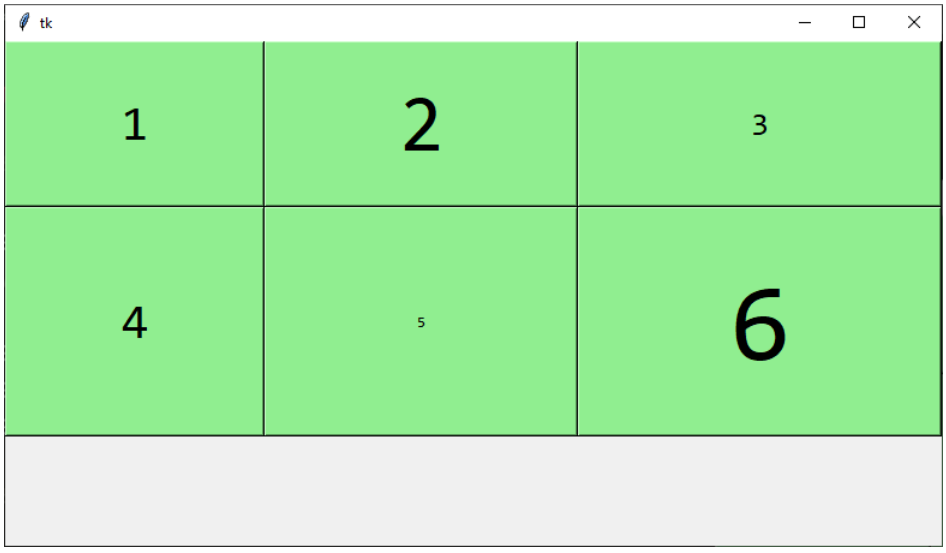
23.- Método grid columnconfigure

Hay una opción que si redimensionamos la ventana los widgets también se redimensionan.

```
5    # ventana.geometry("600x500+500+100")
6    ventana.columnconfigure(0, weight=1)
7    ventana.columnconfigure(1, weight=1)
8    ventana.columnconfigure(2, weight=1)
```

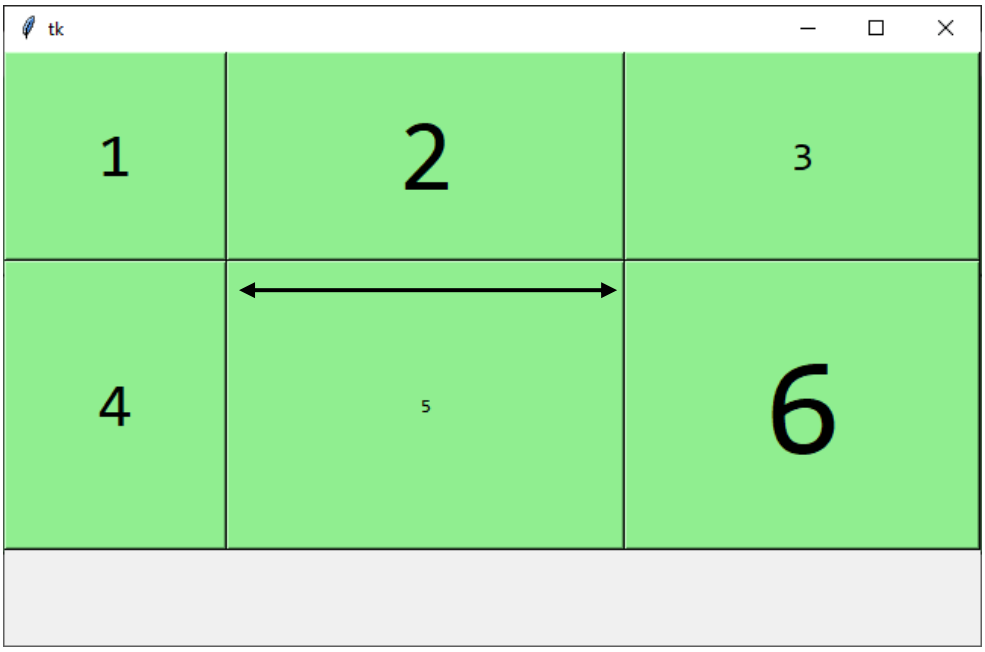
Hemos desactivada la línea 5, ahora le estamos diciendo que las columnas 0, 1 y dos al redimensionar la ventana horizontalmente los widgets se ajustarán a lo acho.





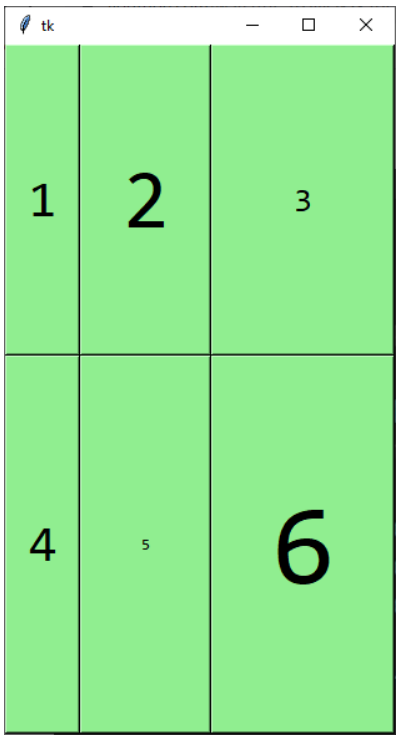

```
5    # ventana.geometry("600x500+500+100")
6    ventana.columnconfigure(0, weight=1)
7    ventana.columnconfigure(1, weight=2)
8    ventana.columnconfigure(2, weight=1)
```

A la columna 2 le hemos dado más peso, esta se redimensionará más.




```
5    # ventana.geometry("600x500+500+100")
6    ventana.columnconfigure(0, weight=1)
7    ventana.columnconfigure(1, weight=1)
8    ventana.columnconfigure(2, weight=1)
9
10   ventana.rowconfigure(0, weight=1)
11   ventana.rowconfigure(1, weight=1)
```

Igual que hemos realizado con las columnas lo podemos realizar con las filas.



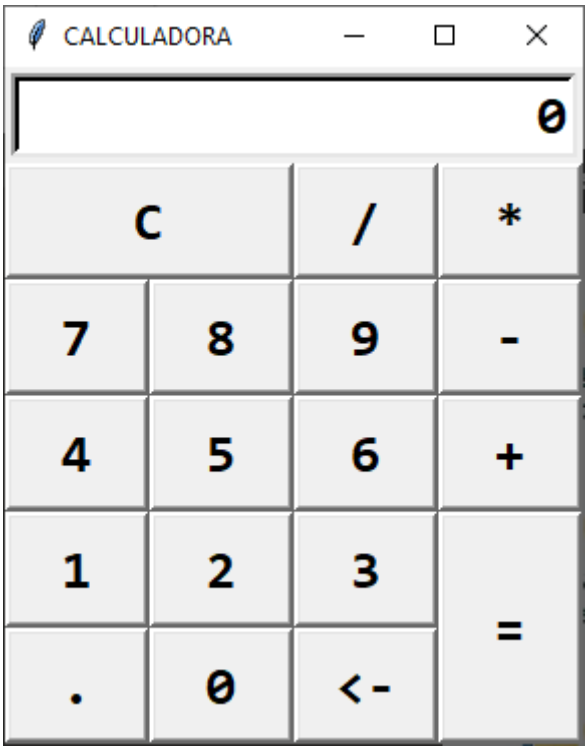
24.- Calculadora con el método grid.

```
import tkinter as tk

##### VENTANA PRINCIPAL #####
ventana= tk.Tk()
ventana.title("CALCULADORA")
ventana.geometry("+600+100")

##### WIDGETS APLICACION #####
pantalla = tk.Label(ventana,
    text="0",
    bg="white",
    font="consolas 20 bold",
    anchor = "e", # Posición en pantalla
    relief="sunken", bd=4)
pantalla.grid(row=0, column=0, columnspan=4, sticky="nsew", padx=3,
pady=3)

limpiar = tk.Button(ventana,
    text="C",
    font="consolas 20 bold",
    relief="raised", bd=3,
    width=4)
limpiar.grid(row=1, column=0, columnspan=2, sticky="nswe")

division = tk.Button(ventana,
    text="/",
    font="consolas 20 bold",
    relief="raised", bd=3,
    width=4)
division.grid(row=1, column=2)

multiplicacion = tk.Button(ventana,
    text="*",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
multiplicacion.grid(row=1, column=3)

siete = tk.Button(ventana,
    text="7",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
siete.grid(row=2, column=0)

ocho = tk.Button(ventana,
    text="8",
```

```
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
ocho.grid(row=2, column=1)

nueve = tk.Button(ventana,
    text="9",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
nueve.grid(row=2, column=2)

resta = tk.Button(ventana,
    text="-",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
resta.grid(row=2, column=3)

cuatro = tk.Button(ventana,
    text="4",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
cuatro.grid(row=3, column=0)

cinco = tk.Button(ventana,
    text="5",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
cinco.grid(row=3, column=1)

seis = tk.Button(ventana,
    text="6",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
seis.grid(row=3, column=2)

sumar = tk.Button(ventana,
    text="+",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
sumar.grid(row=3, column=3)

uno = tk.Button(ventana,
    text="1",
```

```
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
uno.grid(row=4, column=0)

dos = tk.Button(ventana,
    text="2",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
dos.grid(row=4, column=1)

tres = tk.Button(ventana,
    text="3",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
tres.grid(row=4, column=2)

igual = tk.Button(ventana,
    text="=",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
igual.grid(row=4, column=3, rowspan=2, sticky="nsew")

punto = tk.Button(ventana,
    text=".",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
punto.grid(row=5, column=0)

cero = tk.Button(ventana,
    text="0",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
cero.grid(row=5, column=1)

menor = tk.Button(ventana,
    text="<-",
    font="consolas 20 bold",
    relief="raised",bd=3,
    width=4)
menor.grid(row=5, column=2)


##### BUCLE PRINCIPAL #####
```

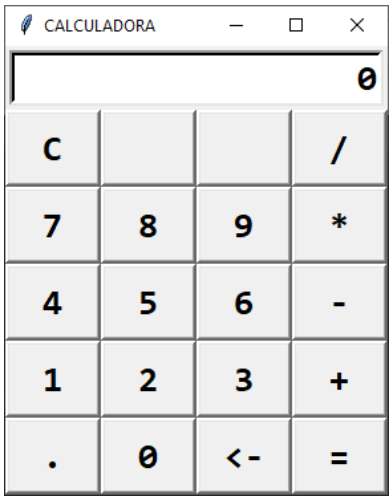
```
ventana.mainloop()
```



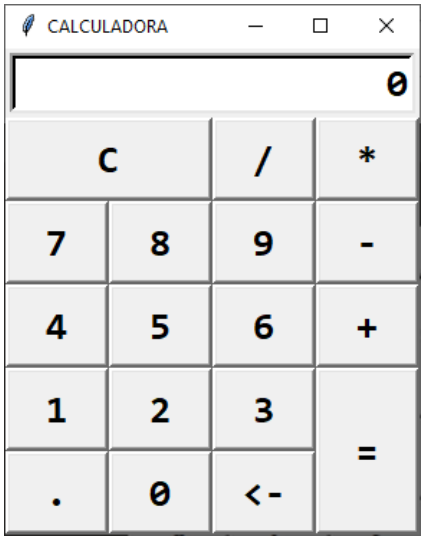
Vamos a dejar por el momento la parte de programación lógica, para realizar otro ejemplo de como diseñar la calculadora con mucha menos líneas de código, para ello vamos a realizar un nuevo proyecto.

```
import tkinter as tk

##### VENTANA PRINCIPAL #####
ventana= tk.Tk()
ventana.title("CALCULADORA")
ventana.geometry("+600+100")

##### WIDGETS APLICACION #####
pantalla = tk.Label(ventana,
    text=0,
    bg="white",
    font="consolas 20 bold",
    anchor = "e",
    relief="sunken", bd=4)
pantalla.grid(row=0, column=0, padx=3, pady=3,
    columnspan=4, sticky="nsew")

teclas = [["C", " ", " ", "/"],
          ["7", "8", "9", "*"],
          ["4", "5", "6", "-"],
          ["1", "2", "3", "+"],
          [".", "0", "<-", "="]]
```

```
for i in range(len(teclas)):
    for j in range(len(teclas[0])):
        tk.Button(ventana,
            text=teclas[i][j],
            font="consolas 20 bold",
            relief="raised", bd=3,
            width=4).grid(row=1+i, column=j)

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```



Vamos a la versión anterior.



Vamos a realizar unas mejoras en su diseño:

```
1    import tkinter as tk
2
```

```
3    ##### VENTANA PRINCIPAL #####
4    ventana= tk.Tk()
5    ventana.title("CALCULADORA")
6    ventana.geometry("+600+100")
7
8    marco = tk.Frame(ventana)
```

Al principio de finimos un nuevo objeto de tipo Frame (marco)

```
145    marco.pack()
146
147    ##### BUCLE PRINCIPAL #####
148    ventana.mainloop()
```

Al final lo insertamos con el método pack().
No podemos mezclar el método grid con el método pack, pero al ser un marco que es donde irán todos los widgets, nos lo va a permitir.
Ahora todos los widget que como origen pone **ventana** los vamos a cambiar por **marco**.

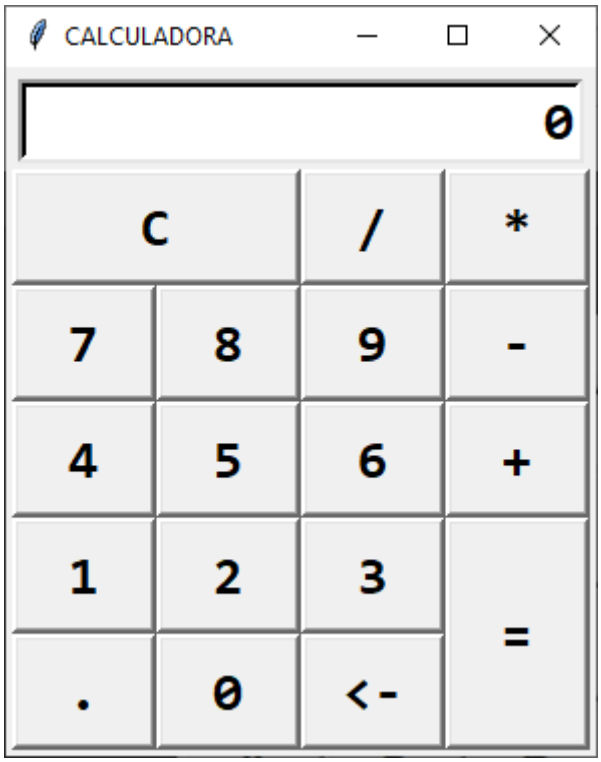
```
10   ##### WIDGETS APLICACION #####
11   pantalla = tk.Label(marco, ←
12       text="0",
13       bg="white",
14       font="consolas 20 bold",
15       anchor = "e", # Posición en pantalla
16       relief="sunken", bd=4)
17   pantalla.grid(row=0, column=0, columnspan=4, sticky="nsew", padx=3, pady=3)
18
19   limpiar = tk.Button(marco,
20       text="C",
21       font="consolas 20 bold",
22       relief="raised", bd=3,
23       width=4)
24   limpiar.grid(row=1, column=0, columnspan=2, sticky="nswe") ←
```

Así con todos los widgets.

```
131  cero = tk.Button(marco, ←
132      text="0",
133      font="consolas 20 bold",
134      relief="raised",bd=3,
135      width=4)
136  cero.grid(row=5, column=1)
137
138  menor = tk.Button(marco, ←
139      text="<-",
140      font="consolas 20 bold",
141      relief="raised",bd=3,
142      width=4)
143  menor.grid(row=5, column=2)
144
145  marco.pack()
```

```
145    marco.pack(padx=3, pady=3)
```

En el marco hacemos que quede un pequeño margen de 3 pixeles.



Además si maximizamos la calculadora quedan en el centro de la pantalla.

25.- Argumentos y funciones mediante funciones lambda

Antes de implementar la funcionalidad de la calculadora vamos a ver un aspecto de tkinter que nos va a facilitar la codificación de las funcionalidades de los botones.

Vamos a realizar un nuevo proyecto:

```
1   import tkinter as tk
2
3   ##### VENTANA PRINCIPAL #####
4   ventana= tk.Tk()
5   ventana.title("CALCULADORA")
6   ventana.geometry("+600+100")
7
8   ##### FUNCIONES SIN ARGUMENTOS #####
9   def pulsar_uno():
10      resultado.set(int(resultado.get())+1)
11
12  def pulsar_cinco():
13      resultado.set(int(resultado.get())+5)
14
15  def pulsar_diez():
16      resultado.set(int(resultado.get())+10)
17
18  resultado = tk.StringVar()
19  resultado.set("0")
20
```

En la línea 18 hemos definido una variable de tipo string para poderla utilizar en widgets.
En la línea 19 le asignamos un valor inicial de “0”.
En la línea 9 hemos creado una función llamada pulsar_uno(): lo que hace es recoger el valor de resultado que tiene en pantalla e incrementarle el valor en un, un contado de 1.
Esto lo repetimos con las funciones de las líneas 12 y 15 pero con incrementos de 5 y 10.

```
21  pantalla = tk.Label(ventana,
22      text="0",
23      textvariable=resultado,
24      bg="white",
25      font="consolas 20 bold",
26      anchor = "e", # Posición en pantalla
27      relief="sunken", bd=4)
28  pantalla.grid(row=0, column=0, columnspan=3, sticky="nsew", padx=3, pady=3)
```

El widget pantalla de tipo tk.Label tenemos una propiedad llamada textvariable que asume el valor que le estamos asignando a la variable resultado que es de tipo tk.StringVar().

```
29
30    uno = tk.Button(ventana,
31        text="1",
32        font="consolas 20 bold",
33        relief="raised", bd=3,
34        width=4,
35        command=pulsar_uno)
36    uno.grid(row=1, column=0, sticky="nswe")
```

En la línea 35 al hacer clic al botón 1 llamamos a la función pulsar_uno.

```
37
38    cinco = tk.Button(ventana,
39        text="5",
40        font="consolas 20 bold",
41        relief="raised", bd=3,
42        width=4,
43        command=pulsar_cinco)
44    cinco.grid(row=1, column=1, sticky="nswe")
```

En la línea 43 al hacer clic en el botón 5 llamamos a la función pulsar_diez.

```
45
46    diez = tk.Button(ventana,
47        text="10",
48        font="consolas 20 bold",
49        relief="raised", bd=3,
50        width=4,
51        command=pulsar_diez)
52    diez.grid(row=1, column=2)
```

En la línea 51 al hacer clic en el botón 10 llamamos a la función pulsar_diez.

```
53
54    ##### BUCLE PRINCIPAL #####
55    ventana.mainloop()
```

Vamos a simplificar codigo, para ello vamos a tener que crear una función en lugar de tres.

Lo primero que vamos a realizar será borrar las tres funciones.

```
8   ##### FUNCIONES CON ARGUMENTOS #####
9
10  def pulsar_numero(numero):
11      resultado.set(int(resultado.get())+numero)
```

En su lugar creamos esta única función.

Y ahora en cada botón:

```
25  uno = tk.Button(ventana,
26      text="1",
27      font="consolas 20 bold",
28      relief="raised", bd=3,
29      width=4,
30      command=lambda: pulsar_numero(1))
31  uno.grid(row=1, column=0, sticky="nswe")
32
33  cinco = tk.Button(ventana,
34      text="5",
35      font="consolas 20 bold",
36      relief="raised", bd=3,
37      width=4,
38      command=lambda: pulsar_numero(5))
39  cinco.grid(row=1, column=1, sticky="nswe")
40
41  diez = tk.Button(ventana,
42      text="10",
43      font="consolas 20 bold",
44      relief="raised", bd=3,
45      width=4,
46      command=lambda: pulsar_numero(10))
47  diez.grid(row=1, column=2)
```

Si omitimos la instrucción lambda, al ejecutar el programa llamará automáticamente a la función una ver y desde el botón este no se podrá llamar.

26.- Funcionalidad de la calculadora de tkinter

Hay una función llamada eval que nos retorna la operación de dos valores más su operador estando estos entre comilllas.

```
>>> eval(“2+3”)
```

```
5
```

```
>>> eval(“45+37/23”)
```

```
46.608695652173914
```

Con esta función nos va a permitir evaluar expresiones matemáticas.

Definimos una variable de tipo String para widgets.

```
10    ##### VARIABLES #####
11    expresion = tk.StringVar()
12    expresion.set("0")
```

Utilizamos al variable en el objeto pantalla de tipo tk.Label:

```
28    ##### WIDGETS APLICACION #####
29    pantalla = tk.Label(marco,
30        textvariable= expresion,
31        bg="white",
32        font="consolas 20 bold",
33        anchor = "e", # Posición en pantalla
34        relief="sunken", bd=4)
35    pantalla.grid(row=0, column=0, columnspan=4, sticky="nsew", padx=3, pady=3)
```

Definimos una función:

```
14    ##### FUNCIONES ######
15    def pulsar_tecla(tecla):
16        actual = expresion.get()
17        if actual == "Error" or actual == "No válido":
18            actual = ""
19        elif actual == "0" and tecla in ("1","2","3","4","5","6","7","8","9","0"):
20            actual = ""
21        elif actual[-1] in ("+", "-", "*", "/") and tecla in ("+", "-", "*", "/"):
22            actual = actual[:-1]
23        actual += tecla
24        expresion.set(actual)
```

En la línea 16 cogemos el valor de la variable de tkinter mediante el método get y asignárselo a la variable actual.

En la línea 17 comprobamos si en la pantalla de nuestra calculadora tenemos “Error” o “No válido” si es así en la línea 18 la variable actual pasaría al valer (vacío).

En la línea 19 si la variable actual es igual a 0 y además un número por ejemplo 05, en la línea 20 la variable actual pasaría a valer (vacío).

En la línea 21 si el último carácter de la expresión es un operador, en la línea 22 eliminaría de la variable actual el último carácter, de este modo no se nos acumularán los operadores.

En la línea 23 a la variable actual de concatenamos le última tecla que hemos presionado.

En la línea 25 pasamos el valor de la variable actual a la variable expresión que si nos acordamos es de tipo tkinter y de este modo se mostrará la información en la pantalla de nuestra calculadora.

Ahora cada vez que pulsemos un número o un operador tiene que llamar a dicha función.

```
division = tk.Button(marco,
    text="/",
    font="consolas 20 bold",
    relief="raised", bd=3,
    command=lambda: pulsar_tecla("/"),
    width=4)
division.grid(row=1, column=2)

multiplicacion = tk.Button(marco,
    text="*",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("*"),
    width=4)
multiplicacion.grid(row=1, column=3)

siete = tk.Button(marco,
    text="7",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("7"),
    width=4)
siete.grid(row=2, column=0)

ocho = tk.Button(marco,
    text="8",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("8"),
    width=4)
ocho.grid(row=2, column=1)

nueve = tk.Button(marco,
    text="9",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("9"),
    width=4)
nueve.grid(row=2, column=2)
```

```
resta = tk.Button(marco,
    text="-",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("-"),
    width=4)
resta.grid(row=2, column=3)

cuatro = tk.Button(marco,
    text="4",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("4"),
    width=4)
cuatro.grid(row=3, column=0)

cinco = tk.Button(marco,
    text="5",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("5"),
    width=4)
cinco.grid(row=3, column=1)

seis = tk.Button(marco,
    text="6",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("6"),
    width=4)
seis.grid(row=3, column=2)

sumar = tk.Button(marco,
    text="+",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("+"),
    width=4)
sumar.grid(row=3, column=3)

punto = tk.Button(marco,
    text=".",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("."),
    width=4)
    punto.grid(row=5, column=0)
```

```
uno = tk.Button(marco,
    text="1",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("1"),
    width=4)
uno.grid(row=4, column=0)

dos = tk.Button(marco,
    text="2",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("2"),
    width=4)
dos.grid(row=4, column=1)

tres = tk.Button(marco,
    text="3",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("3"),
    width=4)
tres.grid(row=4, column=2)

cero = tk.Button(marco,
    text="0",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("0"),
    width=4)
cero.grid(row=5, column=1)
```

Esta es la línea que tienes que agregar. Ahora vamos a crear otra función para cuando tenemos que pulsar el botón = (igual).

```
26    def pulsar_igual():
27        actual = expresion.get()
28        try:
29            resultado = eval(actual)
30        except ZeroDivisionError:
31            resultado = "No válido"
32        except:
33            resultado = "Error"
```

```
34          expresion.set(resultado)
```

En la línea 27 a la variable actual le asignamos el valor que tiene el widget pantalla.

En la línea 28 evitamos que si se produce un error el programa se pueda interrumpir, además de pasar a la variable el mensaje de error, que puede ser “No válido” línea 31.

En la línea 32 si se produce otro tipo de error resultado asuma el valor de Error en la línea 33.

En la línea 34 la variable expresión de tipo tkinter asume el valor de resultado y de este modo se muestra en la pantalla de la calculadora.

Vamos a llamar a la función.

```
156    igual = tk.Button(marco,
157        text="=",
158        font="consolas 20 bold",
159        relief="raised",bd=3,
160        command=pulsar_igual,
161        width=4)
162    igual.grid(row=4, column=3, rowspan=2, sticky="nsew")
```

Vamos a crear la función para limpiar al pantalla.

```
36    def pulsar_limpiar():
37        expresion.set("0")
```

Que será llamada desde:

```
48    limpiar = tk.Button(marco,
49        text="C",
50        font="consolas 20 bold",
51        relief="raised", bd=3,
52        command=pulsar_limpiar,
53        width=4)
54    limpiar.grid(row=1, column=0, columnspan=2, sticky="nswe")
```

Me he dado cuenta que un widget lo había renombrado más vamos a cambiarle el nombre.

```
182    atras = tk.Button(marco,
183        text="<-",
184        font="consolas 20 bold",
185        relief="raised",bd=3,
186        width=4)
187    atras.grid(row=5, column=2)
```

Fíjate que como text pone “<-“ le tenemos que poner el nombre de atras.

Ahora rectificado este pequeño error vamos a realizar la siguiente función:

```
39  def pulsar_atras():
40      actual = expresion.get()
41      if actual == "Error" or actual == "No válido":
42          actual = "0"
43      if len(actual) > 1:
44          actual = actual[:-1]
45          expresion.set(actual)
46      elif len(actual) <= 1:
47          expresion.set("0")
```

En la línea 40 la variable actual asume el valor que hay en pantalla.

En la línea 41 si en pantalla está el mensaje de “Error” o “No válido”, en la línea 42 a la variable actual le asigna el valor de 0.

En la línea 43 si la longitud de la cadena de la variable actual es mayor de 1, en la línea 44 eliminamos el último carácter, puede ser un número o un operador.

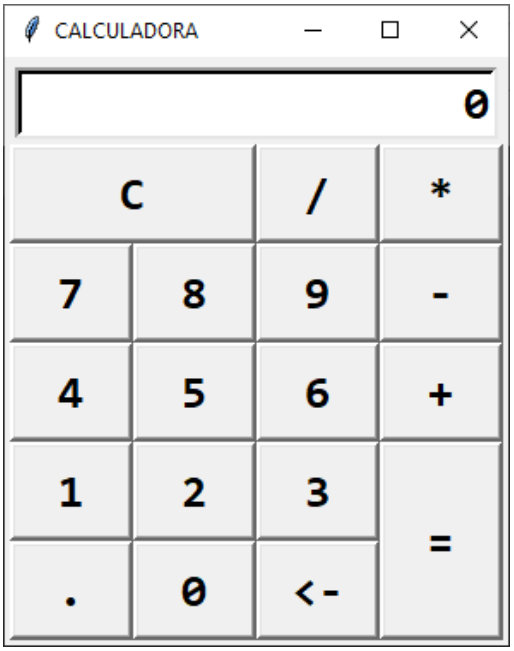
En la línea 45 le pasamos el valor a la variable expresión para que se muestre en pantalla.

En la línea 46 si longitud de la cadena de la variable actual es menor o igual a 1 en la línea 47 a la variable expresión se le asigna el valor de 0 que es lo que se mostrará en pantalla de la calculadora.

Llamaremos esta función desde el botón atrás.

```
195 atras = tk.Button(marco,
196     text="<-",
197     font="consolas 20 bold",
198     relief="raised",bd=3,
199     command=pulsar_atras,
200     width=4)
201 atras.grid(row=5, column=2)
```

Aquí tienes el código completo:




```
import tkinter as tk

##### VENTANA PRINCIPAL #####
ventana= tk.Tk()
ventana.title("CALCULADORA")
ventana.geometry("+600+100")

marco = tk.Frame(ventana)

##### VARIABLES #####
expresion = tk.StringVar()
expresion.set("0")

##### FUNCIONES ######
def pulsar_tecla(tecla):
    actual = expresion.get()
    if actual == "Error" or actual == "No válido":
        actual = ""
    elif actual == "0" and tecla in
("1","2","3","4","5","6","7","8","9","0"):
        actual = ""
    elif actual[-1] in ("+", "-", "*", "/") and tecla in ("+", "-", "*",
"/"):
        actual = actual[:-1]
    actual += tecla
    expresion.set(actual)

def pulsar_igual():
    actual = expresion.get()
    try:
        resultado = eval(actual)
    except ZeroDivisionError:
        resultado = "No válido"
    except:
        resultado = "Error"
    expresion.set(resultado)

def pulsar_limpiar():
    expresion.set("0")

def pulsar_atras():
    actual = expresion.get()
    if actual == "Error" or actual == "No válido":
        actual = "0"
    if len(actual) > 1:
        actual = actual[:-1]
        expresion.set(actual)
```

```
    elif len(actual) <= 1:
        expresion.set("0")

##### WIDGETS APLICACION #####
pantalla = tk.Label(marco,
    textvariable= expresion,
    bg="white",
    font="consolas 20 bold",
    anchor = "e", # Posición en pantalla
    relief="sunken", bd=4)
pantalla.grid(row=0, column=0, columnspan=4, sticky="nsew", padx=3,
pady=3)

limpiar = tk.Button(marco,
    text="C",
    font="consolas 20 bold",
    relief="raised", bd=3,
    command=pulsar_limpiar,
    width=4)
limpiar.grid(row=1, column=0, columnspan=2, sticky="nswe")

division = tk.Button(marco,
    text="/",
    font="consolas 20 bold",
    relief="raised", bd=3,
    command=lambda: pulsar_tecla("/"),
    width=4)
division.grid(row=1, column=2)

multiplicacion = tk.Button(marco,
    text="*",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("*"),
    width=4)
multiplicacion.grid(row=1, column=3)

siete = tk.Button(marco,
    text="7",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("7"),
    width=4)
siete.grid(row=2, column=0)

ocho = tk.Button(marco,
    text="8",
    font="consolas 20 bold",
    relief="raised",bd=3,
```

```
    command=lambda: pulsar_tecla("8"),
    width=4)
ocho.grid(row=2, column=1)

nueve = tk.Button(marco,
    text="9",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("9"),
    width=4)
nueve.grid(row=2, column=2)

resta = tk.Button(marco,
    text="-",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("-"),
    width=4)
resta.grid(row=2, column=3)

cuatro = tk.Button(marco,
    text="4",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("4"),
    width=4)
cuatro.grid(row=3, column=0)

cinco = tk.Button(marco,
    text="5",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("5"),
    width=4)
cinco.grid(row=3, column=1)

seis = tk.Button(marco,
    text="6",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("6"),
    width=4)
seis.grid(row=3, column=2)

sumar = tk.Button(marco,
    text="+",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("+"),
```

```
    width=4)
sumar.grid(row=3, column=3)

uno = tk.Button(marco,
    text="1",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("1"),
    width=4)
uno.grid(row=4, column=0)

dos = tk.Button(marco,
    text="2",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("2"),
    width=4)
dos.grid(row=4, column=1)

tres = tk.Button(marco,
    text="3",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("3"),
    width=4)
tres.grid(row=4, column=2)

igual = tk.Button(marco,
    text="=",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=pulsar_igual,
    width=4)
igual.grid(row=4, column=3, rowspan=2, sticky="nsew")

punto = tk.Button(marco,
    text=".",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("."),
    width=4)
punto.grid(row=5, column=0)

cero = tk.Button(marco,
    text="0",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=lambda: pulsar_tecla("0"),
    width=4)
```

```
cero.grid(row=5, column=1)

atras = tk.Button(marco,
    text="<-",
    font="consolas 20 bold",
    relief="raised",bd=3,
    command=pulsar_atras,
    width=4)
atras.grid(row=5, column=2)

marco.pack(padx=3, pady=3)

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

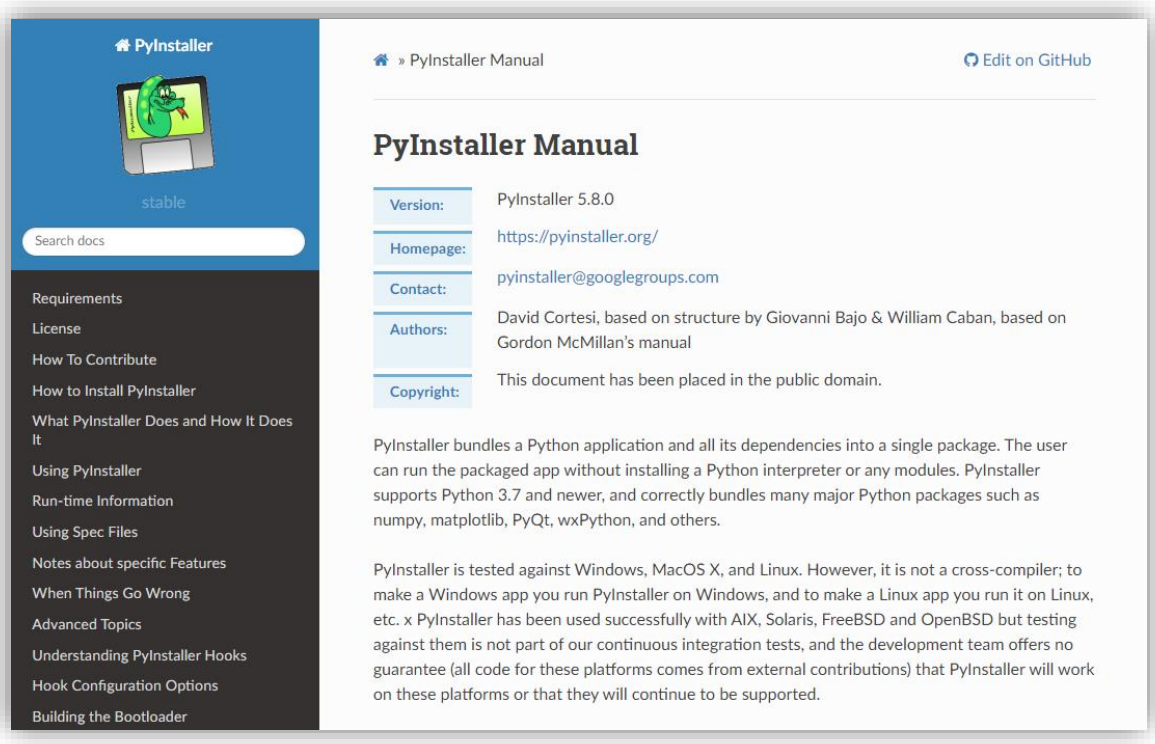
27.- Convertir un archivo .py a un archivo .exe

En el capítulo anterior terminamos de llevar a cabo la aplicación calculadora y podría ser que quisiéramos dar a uno el uso de esta aplicación u otra y para facilitar su uso y no tener que abrir cada vez el archivo de Python y ejecutarlo pues podría ser que prefiriésemos que tenerlo compilado en un archivo exe.

Vamos a ver como podemos convertir en archivo en exe.

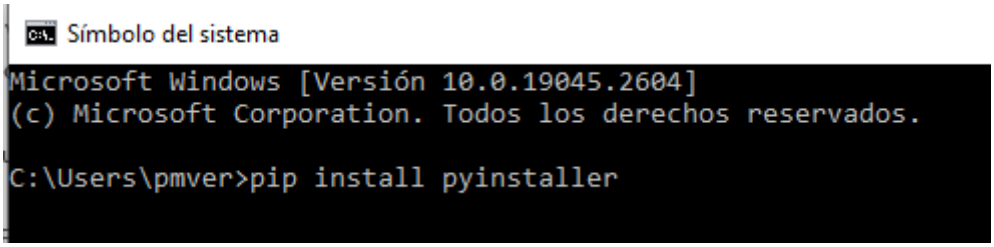
Vamos a tener que instalar un módulo llamado PyInstaler, desde la siguiente url.

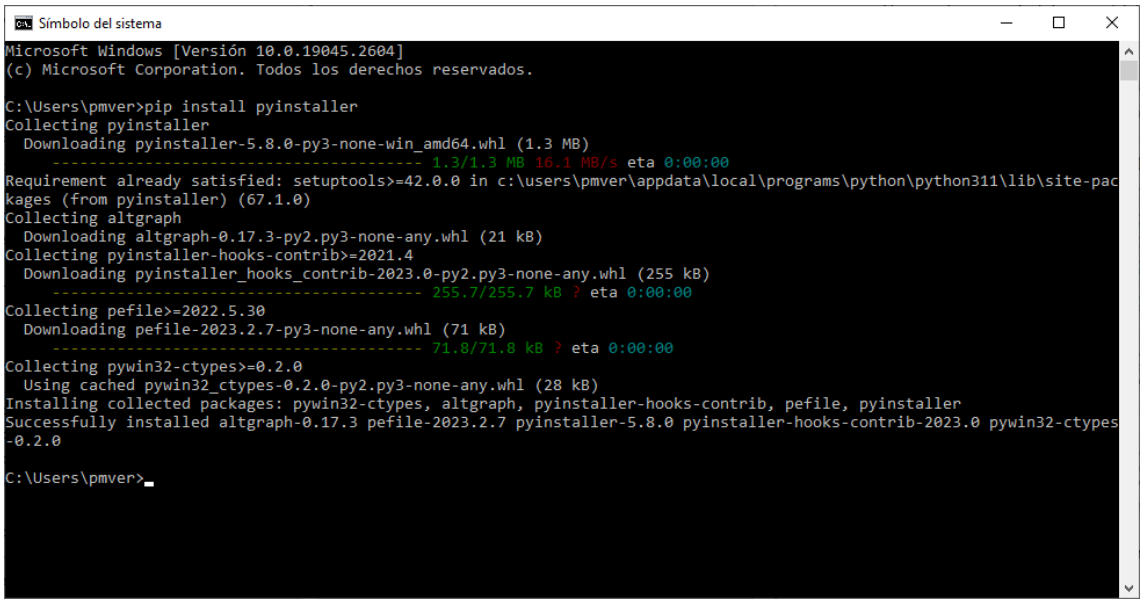
https://pyinstaller.org/en/stable/



También lo podemos instalar con la herramienta pip, desde Cmd.

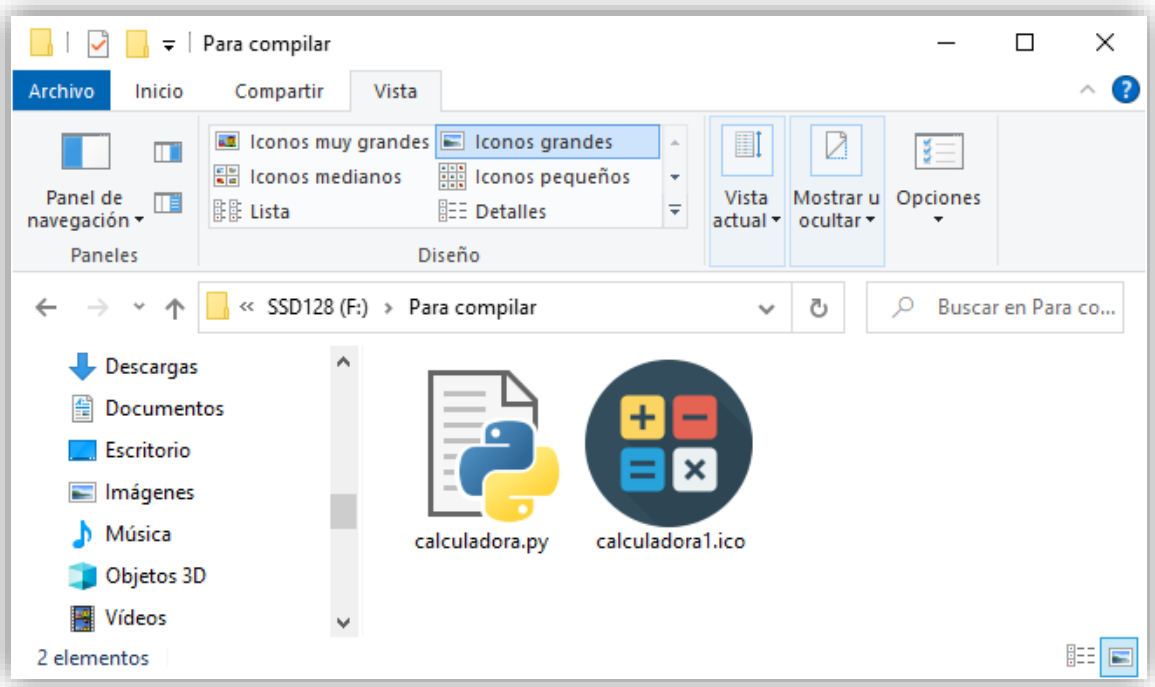
pip install pyinstaller



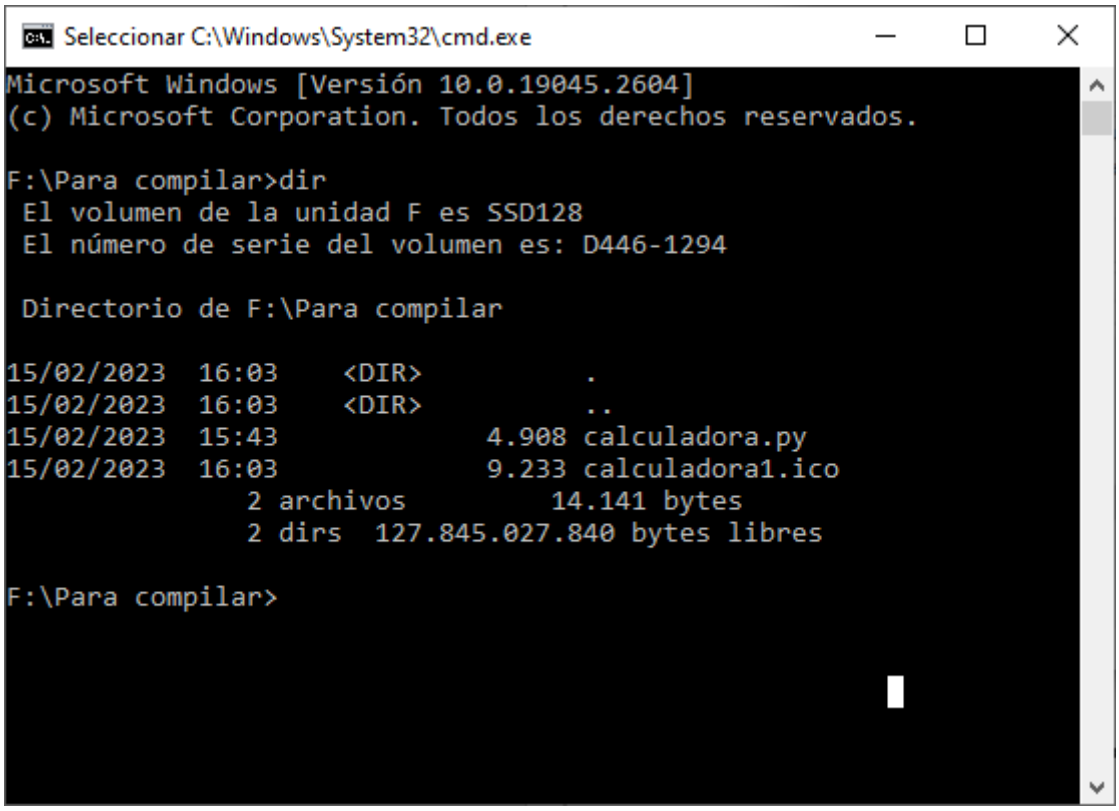


Ya lo hemos instalado.

Ahora en la capeta que tenemos la calculadora le agregaremos un archivo de tipo ico.



Desde la ventana de Cmd vamos a dicha carpeta.



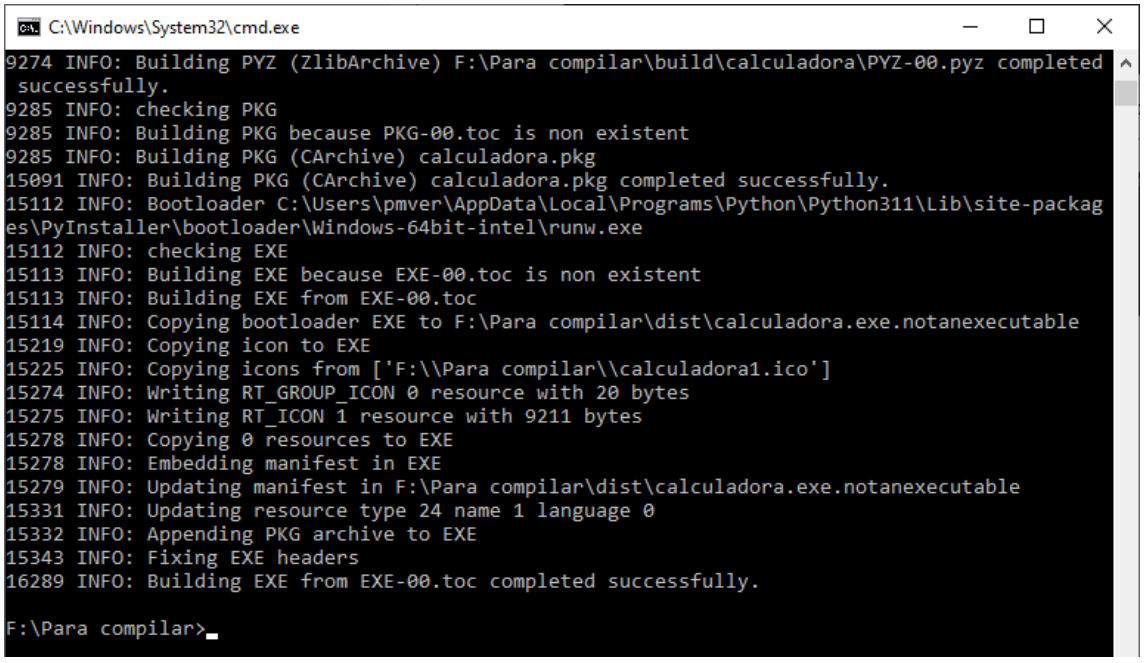
C:\> pyinstaller - -onefile - -windowed - -icon=calculadora1.ico calculadora.py

onefiler → Si puede que lo compile en un solo archivo.

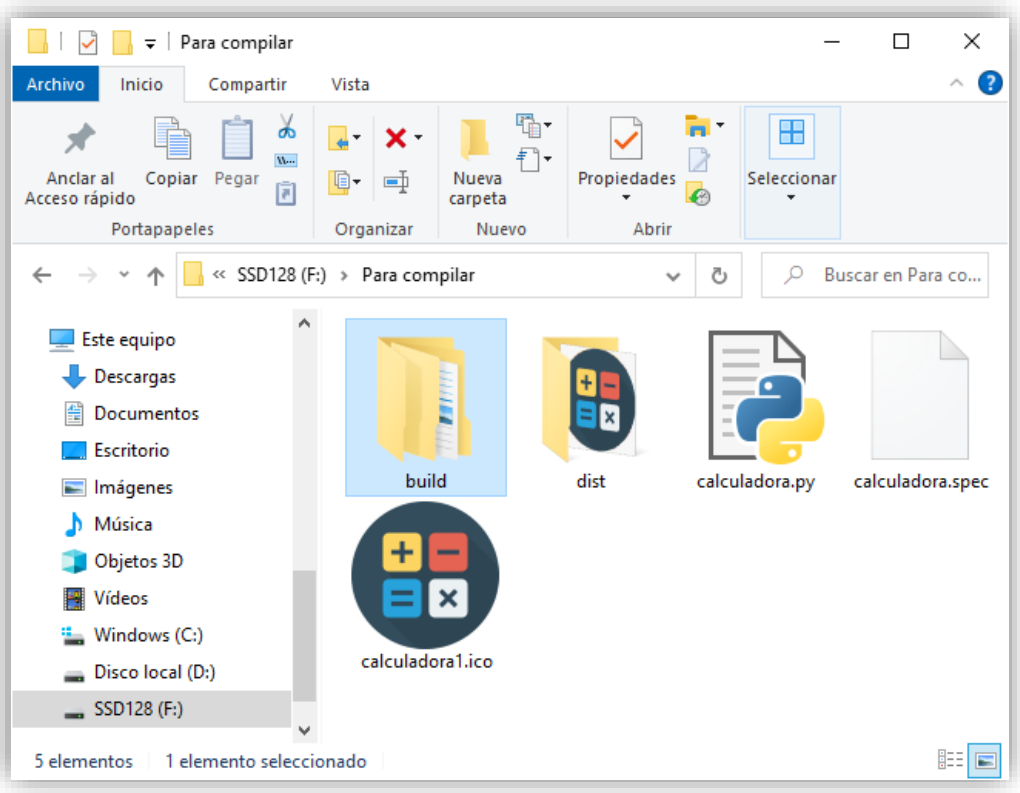
Windowed → Al ejecutar que no muestre la consola.

--icon →Para agregar el archivo de tipo ico.

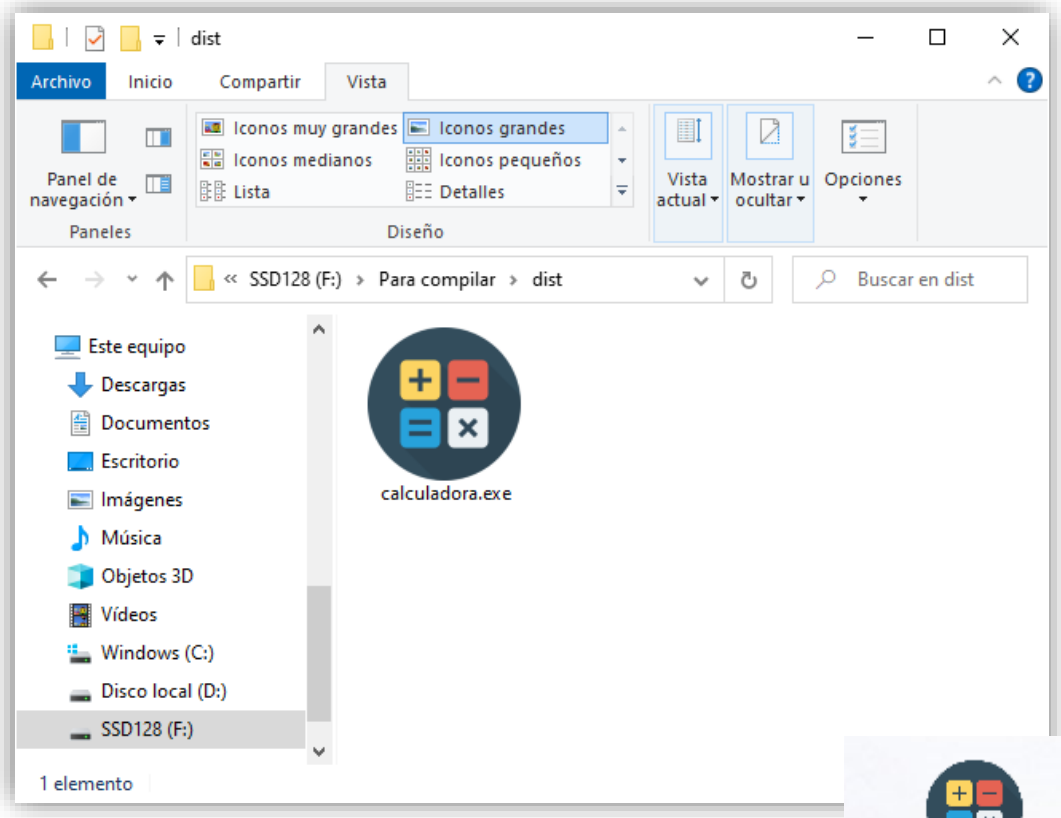
Por último el proyecto realizado en Python.



Vamos a la carpeta:



Entramos en la carpeta dist.

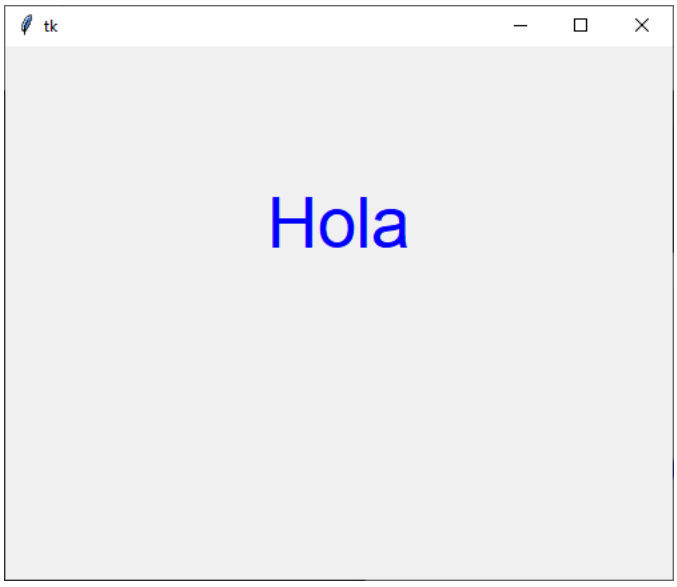


Este archivo lo puede arrastrar a donde tú quieras y a ejecutarlo.

28.- Método after de los widgets en tkinter

En este capítulo vamos a aprender a establecer temporizadores para nuestras aplicaciones Tkinter.

```
1   import tkinter as tk
2
3   ##### VENTANA PRINCIPAL #####
4   ventana = tk.Tk()
5   ventana.geometry("500x400")
6
7   ##### FUNCIONES #####
8
9
10  ##### WIDGETS #####
11  palabra = tk.StringVar()
12  palabra.set("Hola")
13  rotulo = tk.Label(ventana,
14      textvariable = palabra,
15      font = ("Arial 40"),
16      fg="blue")
17  rotulo.pack(padx=50, pady=100)
18
19  ##### BUCLE PRINCIPAL #####
20  ventana.mainloop()
```



Vamos a realizar la siguiente función:

```
7   ##### FUNCIONES #####
8   def actualizar():
9       if palabra.get() == "Hola":
10          palabra.set("")
11      else:
12          palabra.set("Hola")
```

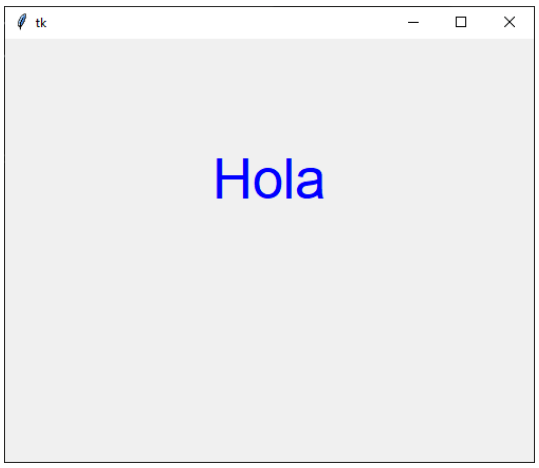
A continuación al final del código antes de llegar al bucle llamamos a la función.

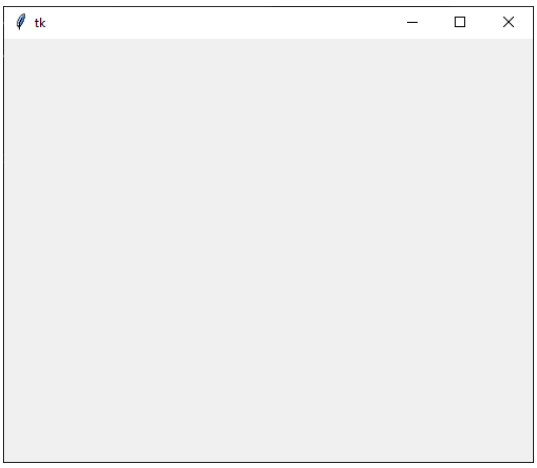
```
24  rotulo.after(1000, actualizar)
```

Ahora cuando ejecutemos la función esta solo se ejecutará una vez, para que repita más veces, modificaremos la función actulizar().

```
7   ##### FUNCIONES #####
8   def actualizar():
9       if palabra.get() == "Hola":
10          palabra.set("")
11      else:
12          palabra.set("Hola")
13      rotulo.after(1000, actualizar)
```

Agregaremos la última línea.





Se mostrará el texto intermitentemente por periodos de 1 segundo.

```python
import tkinter as tk

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.geometry("500x400")

##### FUNCIONES #####
def actualizar():
    if palabra.get() == "Hola":
        palabra.set("")
    else:
        palabra.set("Hola")
    ventana.after(1000, actualizar)

##### WIDGETS #####
palabra = tk.StringVar()
palabra.set("Hola")
rotulo = tk.Label(ventana,
    textvariable = palabra,
    font = ("Arial 40"),
    fg="blue")
rotulo.pack(padx=50, pady=100)

ventana.after(1000, actualizar)

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

Podemos cambiar rotulo por ventana y la aplicación funciona igual.

29.- Método after_cancel

En el capítulo anterior ya vimos como podíamos establecer temporizadores para aplicaciones de Tkinter, vamos a ver más aspectos que nos van a permitir más opciones del método after.

```
1   import tkinter as tk
2   ##### VENTANA PRINCIPAL #####
3   ventana = tk.Tk()
4   ventana.geometry("500x400")
5   ##### FUNCIONES #####
6   contador = 0
7
8   def actualizar(contador):
9       contador += 1
10      palabra.set(contador)
11      tarea = rotulo.after(1000, actualizar, contador)
12      if contador >= 10:
13          rotulo.after_cancel(tarea)
14          palabra.set("Fin")
15
16  ##### WIDGETS #####
17  palabra = tk.StringVar()
18  palabra.set(contador)
19  rotulo = tk.Label(ventana,
20      textvariable = palabra,
21      font = ("Arial 40"),
22      fg="blue")
23  rotulo.pack(padx=50, pady=100)
24
25  rotulo.after(1000, actualizar, contador)
26  ##### BUCLE PRINCIPAL #####
27  ventana.mainloop()
```

La línea 6 definimos a la variable contado a 0.

La línea 25 llama a la funciona actualizar que empieza en la línea 8.

En la función en la línea 9 la variable contador incrementa en 1.

En la línea 10 muestra el valor de contador en el widget rotulo.

En la línea 11 la variable tarea es igual a la llamada a la función de nuevo.

En la línea 12 si contador en mayor o igual a 10, en la línea 13 se cancela la llamada a la función y en la línea 14 muestra el mensaje Fin, después de haber contado del 1 hasta el 9.

30.- Reto 30 – Widget Notebook

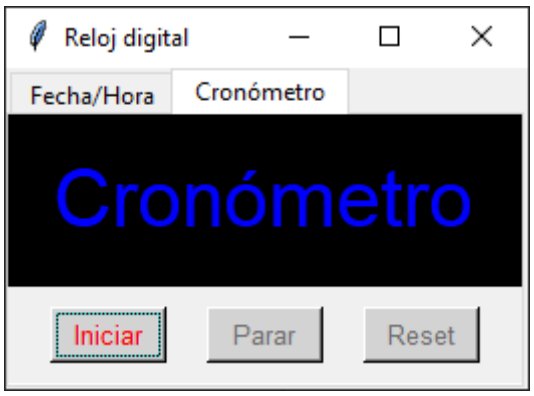
Este proyecto consta de dos pestañas, una para Fecha/hora y otra para Cronómetro.

También tenemos un menú Modos, con varias opciones:


```
import tkinter as tk
from tkinter import ttk

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.title("Reloj digital")
ventana.geometry("260x160+900+100")
```

```
##### CREACIÓN DEL CUADERNO #####
cuaderno = ttk.Notebook(ventana)
cuaderno.pack(expand = True, fill=tk.BOTH)

##### PAGINA 1 DEL CUADERNO #####
pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
cuaderno.add(pagina_1, text="  Fecha/Hora  ")

rotulo_fecha = tk.Label(pagina_1,
    text="Fecha",
    font="arial 18 bold",
    bg="black", fg="green")
rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)

rotulo_hora = tk.Label(pagina_1,
    text="Hora",
    font="arial 36 bold",
    bg="black", fg="blue",
    anchor=tk.N)
rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)

##### PAGINA 2 DEL CUADERNO #####

pagina_2 = tk.Frame(cuaderno)
cuaderno.add(pagina_2, text="  Cronómetro  ")

rotulo_cronometro = tk.Label(pagina_2,
    text="Cronómetro",
    font="arial 30",
    bg="black", fg="blue")
rotulo_cronometro.pack(expand=True, fill=tk.BOTH)

frame_botones = tk.Frame(pagina_2)

boton_iniciar = tk.Button(frame_botones,
    font="arial 10",
    text="Iniciar",
    width=6,
    bg="lightgrey", fg="red")
boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

boton_parar = tk.Button(frame_botones,
    font="arial 10",
    text="Parar",
    width=6,
    state="disabled",
    bg="lightgrey", fg="red")
boton_parar.pack(padx=10, pady=10, side=tk.LEFT)
```

```
boton_resetear = tk.Button(frame_botones,
    font="arial 10",
    text="Reset",
    width=6,
    state="disabled",
    bg ="lightgrey", fg="red")
boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)

frame_botones.pack()

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```



Vamos a comentar parte del código:

```
2    from tkinter import ttk
```

Para poder agregar un widget Notebook necesitamos importar esta libreriá.

```
 9    ##### CREACIÓN DEL CUADERNO #####
10    cuaderno = ttk.Notebook(ventana)
11    cuaderno.pack(expand = True, fill=tk.BOTH)
```

Creamos un objeto llamado cuaderno de la librería ttk de tipo notebook que se insertará en nuestra ventana.

Este objeto se adaptarla al tamaño de la ventana.

```
13    ##### PAGINA 1 DEL CUADERNO #####
14    pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
15    cuaderno.add(pagina_1, text="  Fecha/Hora  ")
```

El objeto pagina_1 es para crear la primera pestaña del proyecto. Se inserta sobre el objeto cuaderno y tendrá un texto de “Fecha/Hora”.

```
17    rotulo_fecha = tk.Label(pagina_1,
18        text="Fecha",
19        font="arial 18 bold",
20        bg="black", fg="green")
21    rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)
22
23    rotulo_hora = tk.Label(pagina_1,
24        text="Hora",
25        font="arial 36 bold",
26        bg="black", fg="blue",
27        anchor=tk.N)
28    rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
```

Insertamos las dos etiquetas:

```
30    ##### PAGINA 2 DEL CUADERNO #####
31    pagina_2 = tk.Frame(cuaderno)
32    cuaderno.add(pagina_2, text="  Cronómetro  ")
```

El objeto pagina_2 es para crear la segunda pestaña del proyecto. Se inserta sobre el objeto cuaderno y tendrá un texto de “Cronómetro”.

```
34    rotulo_cronometro = tk.Label(pagina_2,
35        text="Cronómetro",
36        font="arial 30",
37        bg="black", fg="blue")
38    rotulo_cronometro.pack(expand=True, fill=tk.BOTH)
39
40    frame_botones = tk.Frame(pagina_2)
41
42    boton_iniciar = tk.Button(frame_botones,
43        font="arial 10",
44        text="Iniciar",
45        width=6,
46        bg="lightgrey", fg="red")
47    boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)
48
49    boton_parar = tk.Button(frame_botones,
50        font="arial 10",
51        text="Parar",
52        width=6,
53        state="disabled",
54        bg="lightgrey", fg="red")
55    boton_parar.pack(padx=10, pady=10, side=tk.LEFT)
56
57    boton_resetear = tk.Button(frame_botones,
58        font="arial 10",
59        text="Reset",
60        width=6,
61        state="disabled",
62        bg ="lightgrey", fg="red")
63    boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)
64
```

Insertamos una etiqueta, un marco para los botones y tres botones.

```
65    frame_botones.pack()
```

Añadimos el marco.

```
67    ##### BUCLE PRINCIPAL #####
68    ventana.mainloop()
```

El bucle final.

31.- Aplicación de reloj digital

En este capítulo vamos a dar funcionalidad a la fecha y a la hora.

```
3    import time
```

Importamos la librería time.

```
10   ##### FUNCIONES RELOJ #####
11   def actualizar_hora():
12       hora.set(time.strftime("%H:%M:%S"))
13       rotulo_hora.after(500, actualizar_hora)
14       rotulo_hora.focus()
15
16   def actualizar_fecha():
17       fecha.set(time.strftime("%d / %m / %Y"))
18       rotulo_fecha.after(500, actualizar_fecha)
```

Creamos las funciones para actualizar la hora y actualizar la fecha, con sus respectivos formatos, que se irá actualizando cada 500 segundos.

```
28   fecha = tk.StringVar()
29
30   rotulo_fecha = tk.Label(pagina_1,
31       textvariable = fecha,
32       font="arial 18 bold",
33       bg="black", fg="green")
34   rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)
35
36   hora = tk.StringVar()
37
38   rotulo_hora = tk.Label(pagina_1,
39       textvariable = hora,
40       font="arial 36 bold",
41       bg="black", fg="blue",
42       anchor=tk.N)
43   rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
```

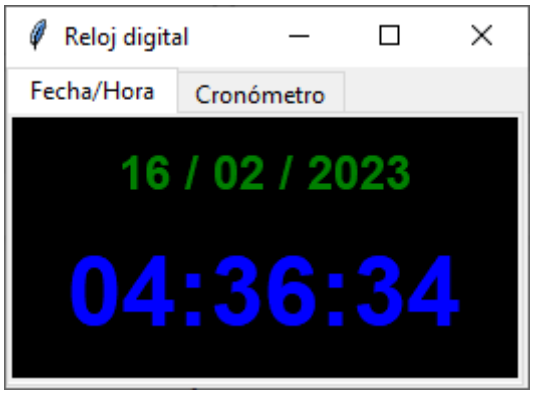
Creamos las variables fecha y hora de tipo tkinter.

Cambiamos la propiedad text por textvariables y le asignamos su correspondiente variable.

```
45   actualizar_hora()
46   actualizar_fecha()
```

Llamamos por primera ver la función actualizar_hora() y actualizar_fecha().

Luego las mismas funciones se llamarán cada 500 segundos.



Vamos por la parte de cronómetro:

```
20  ##### FUNCIONES CRONOMETRO #####
21  horas, minutos,segundos = 0, 0, 0
22  tarea = None
23
24  def iniciar_cronometro():
25      global horas, minutos, segundos
26      global tarea
27
28      tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
29      segundos += 1
30      if segundos == 60:
31          segundos = 0
32          minutos += 1
33      if minutos == 60:
34          minutos = 0
35          horas += 1
36      tarea = rotulo_cronometro.after(1000, iniciar_cronometro)
37
38      rotulo_cronometro.focus()
39      boton_iniciar["state"] = "disabled"
40      boton_resetear["state"] = "disabled"
41      boton_parar["state"] = "normal"
42
43  def parar_cronometro():
44      global tarea
45
46      rotulo_cronometro.after_cancel(tarea)
47      boton_iniciar["state"] = "normal"
48      boton_resetear["state"] = "normal"
49      boton_parar["state"] = "disabled"
50
```

```
51  def resetear_cronometro():
52      global horas, minutos, segundos
53      horas, minutos,segundos = 0, 0, 0
54      tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
55
56      boton_iniciar["state"] = "normal"
57      boton_resetear["state"] = "disabled"
58      boton_parar["state"] = "disabled"
```

Hemos creado las tres funciones que irán a sus respectivos botones para ser llamadas.

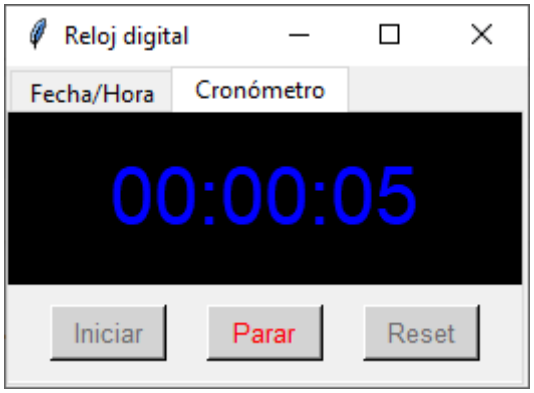
```
94  tiempo = tk.StringVar()
95  tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
96
97  rotulo_cronometro = tk.Label(pagina_2,
98      textvariable = tiempo,
99      font="arial 30",
100     bg="black", fg="blue")
101 rotulo_cronometro.pack(expand=True, fill=tk.BOTH)
```

Definimos la variable tiempo de tipo tkinter, al objeto rotulo_cronometro le asignamos a la propiedad textvariable.

```
105 boton_iniciar = tk.Button(frame_botones,
106     font="arial 10",
107     text="Iniciar",
108     width=6,
109     bg="lightgrey", fg="red",
110     command=iniciar_cronometro)
111 boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)
112
113 boton_parar = tk.Button(frame_botones,
114     font="arial 10",
115     text="Parar",
116     width=6,
117     state="disabled",
118     bg="lightgrey", fg="red",
119     command=parar_cronometro)
120 boton_parar.pack(padx=10, pady=10, side=tk.LEFT)
121
122 boton_resetear = tk.Button(frame_botones,
123     font="arial 10",
124     text="Reset",
```

```
125         width=6,
126         state="disabled",
127         bg ="lightgrey", fg="red",
128         command=resetear_cronometro )
129     boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)
```

Desde los respectivos botones llamamos a las funciones.



Adjunto de nuevo todo el código por si tuvieras que revisarlo:

```
import tkinter as tk
from tkinter import ttk
import time

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.title("Reloj digital")
ventana.geometry("260x160+900+100")

##### FUNCIONES RELOJ #####
def actualizar_hora():
    hora.set(time.strftime("%H:%M:%S"))
    rotulo_hora.after(500, actualizar_hora)
    rotulo_hora.focus()

def actualizar_fecha():
    fecha.set(time.strftime("%d / %m / %Y"))
    rotulo_fecha.after(500, actualizar_fecha)

##### FUNCIONES CRONOMETRO #####
horas, minutos,segundos = 0, 0, 0
tarea = None

def iniciar_cronometro():
    global horas, minutos, segundos
    global tarea

    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
```

```
        segundos += 1
        if segundos == 60:
            segundos = 0
            minutos += 1
        if minutos == 60:
            minutos = 0
            horas += 1
        tarea = rotulo_cronometro.after(1000, iniciar_cronometro)

        rotulo_cronometro.focus()
        boton_iniciar["state"] = "disabled"
        boton_resetear["state"] = "disabled"
        boton_parar["state"] = "normal"

    def parar_cronometro():
        global tarea

        rotulo_cronometro.after_cancel(tarea)
        boton_iniciar["state"] = "normal"
        boton_resetear["state"] = "normal"
        boton_parar["state"] = "disabled"

    def resetear_cronometro():
        global horas, minutos, segundos
        horas, minutos,segundos = 0, 0, 0
        tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

        boton_iniciar["state"] = "normal"
        boton_resetear["state"] = "disabled"
        boton_parar["state"] = "disabled"



    ##### CREACIÓN DEL CUADERNO #####
    cuaderno = ttk.Notebook(ventana)
    cuaderno.pack(expand = True, fill=tk.BOTH)

    ##### PAGINA 1 DEL CUADERNO #####
    pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
    cuaderno.add(pagina_1, text="  Fecha/Hora  ")

    fecha = tk.StringVar()

    rotulo_fecha = tk.Label(pagina_1,
        textvariable = fecha,
        font="arial 18 bold",
        bg="black", fg="green")
    rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)
```

```
hora = tk.StringVar()

rotulo_hora = tk.Label(pagina_1,
    textvariable = hora,
    font="arial 36 bold",
    bg="black", fg="blue",
    anchor=tk.N)
rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)

actualizar_hora()
actualizar_fecha()

##### PAGINA 2 DEL CUADERNO #####
pagina_2 = tk.Frame(cuaderno)
cuaderno.add(pagina_2, text="  Cronómetro  ")

tiempo = tk.StringVar()
tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

rotulo_cronometro = tk.Label(pagina_2,
    textvariable = tiempo,
    font="arial 30",
    bg="black", fg="blue")
rotulo_cronometro.pack(expand=True, fill=tk.BOTH)

frame_botones = tk.Frame(pagina_2)

boton_iniciar = tk.Button(frame_botones,
    font="arial 10",
    text="Iniciar",
    width=6,
    bg="lightgrey", fg="red",
    command=iniciar_cronometro)
boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

boton_parar = tk.Button(frame_botones,
    font="arial 10",
    text="Parar",
    width=6,
    state="disabled",
    bg="lightgrey", fg="red",
    command=parar_cronometro)
boton_parar.pack(padx=10, pady=10, side=tk.LEFT)

boton_resetear = tk.Button(frame_botones,
    font="arial 10",
    text="Reset",
    width=6,
    state="disabled",
```

```
    bg ="lightgrey", fg="red",
    command=resetear_cronometro )
boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)

frame_botones.pack()

##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

32.- Widget Menu

En este capítulo vamos a agregar un menú de opciones a nuestro cronómetro.

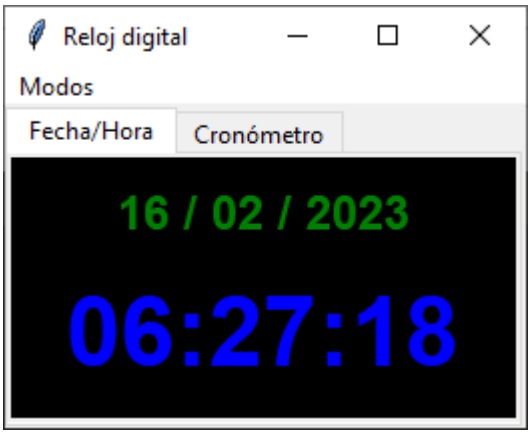
Vamos a crear el menú:

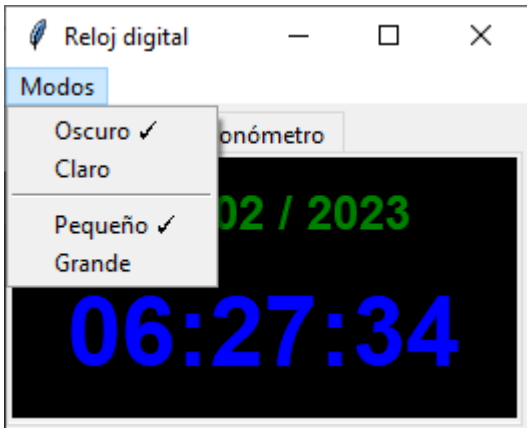
```
168   ##### CREACIÓN DE LOS MENUS #####
169   # Creación de la barra de menús
170   barra_menus = tk.Menu(ventana)
171
172   # Cración de un menú
173   menu_modos = tk.Menu(barra_menus, tearoff=0)
174
175   # Creación de las opciones de un menú
176   menu_modos.add_command(label="Oscuro \u2713", command=modo_oscuro)
177   menu_modos.add_command(label="Claro", command=modo_claro)
178   menu_modos.add_separator()
179   menu_modos.add_command(label="Pequeño \u2713", command=modo_pequenio)
180   menu_modos.add_command(label="Grande", command=modo_grande)
181
182   # Colocamos el menú en la barra de menús
183   barra_menus.add_cascade(label="Modos", menu=menu_modos)
184
185   # Colocamos la barra de menús en la ventana
186   ventana.config(menu=barra_menus)
```

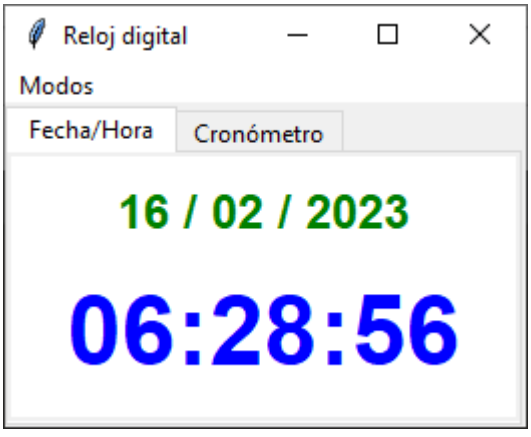
Que llamará a las correspondientes funciones:

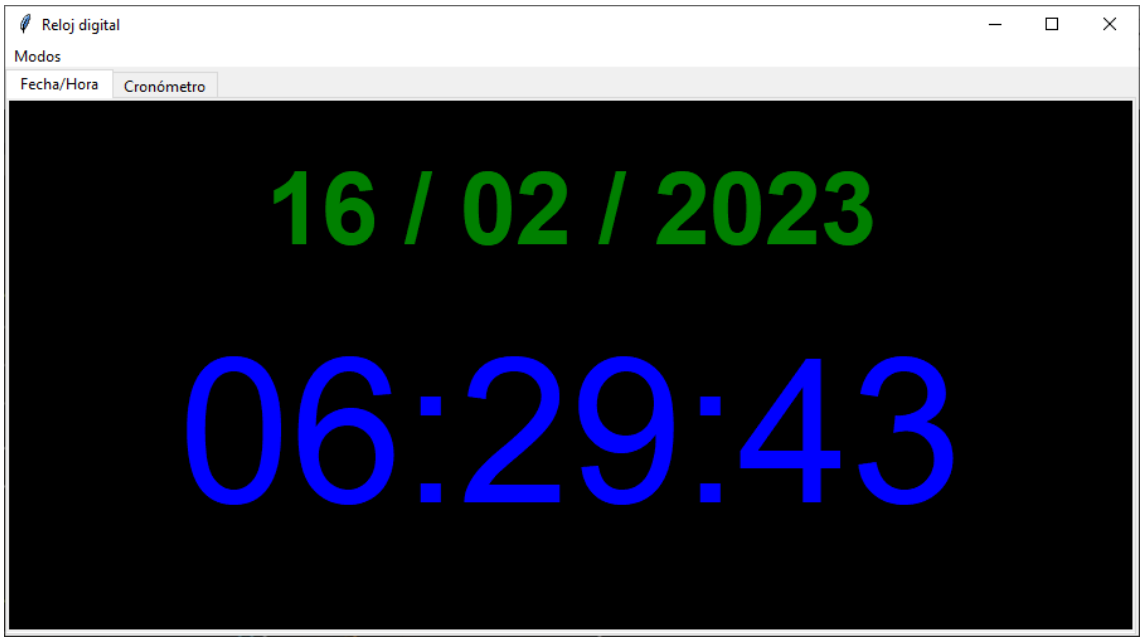
```
60    ##### FUNCIONES MENUS #####
61    def modo_oscuro():
62        rotulo_fecha.config(background="black")
63        rotulo_hora.config(background="black")
64        rotulo_cronometro.config(background="black")
65        menu_modos.entryconfig(0, label="Oscuro \u2713")
66        menu_modos.entryconfig(1, label="Claro")
67
68    def modo_claro():
69        rotulo_fecha.config(background="white")
70        rotulo_hora.config(background="white")
71        rotulo_cronometro.config(background="white")
72        menu_modos.entryconfig(0, label="Oscuro")
73        menu_modos.entryconfig(1, label="Claro \u2713")
74
```

```
75  def modo_pequenio():
76      ventana.geometry("260x160+500+100")
77      rotulo_fecha.config(font="arial 18 bold")
78      rotulo_hora.config(font="arial 36")
79      rotulo_cronometro.config(font="arial 30")
80      boton_iniciar.config(font="arial 10")
81      boton_parar.config(font="arial 10")
82      boton_resetear.config(font="arial 10")
83      menu_modos.entryconfig(3, label="Pequeño \u2713")
84      menu_modos.entryconfig(4, label="Grande")
85
86  def modo_grande():
87      ventana.geometry("900x450+300+100")
88      rotulo_fecha.config(font="arial 60 bold")
89      rotulo_hora.config(font="arial 120")
90      rotulo_cronometro.config(font="arial 90")
91      boton_iniciar.config(font="arial 30")
92      boton_parar.config(font="arial 30")
93      boton_resetear.config(font="arial 30")
94      menu_modos.entryconfig(3, label="Pequeño")
95      menu_modos.entryconfig(4, label="Grande \u2713")
```









Adjunto todo el código para posibles consulta de dudas:

```
import tkinter as tk
from tkinter import ttk
import time

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.title("Reloj digital")
ventana.geometry("260x160+900+100")

##### FUNCIONES RELOJ #####
def actualizar_hora():
    hora.set(time.strftime("%H:%M:%S"))
    rotulo_hora.after(500, actualizar_hora)
```

```
    rotulo_hora.focus()

def actualizar_fecha():
    fecha.set(time.strftime("%d / %m / %Y"))
    rotulo_fecha.after(500, actualizar_fecha)

##### FUNCIONES CRONOMETRO #####
horas, minutos,segundos = 0, 0, 0
tarea = None

def iniciar_cronometro():
    global horas, minutos, segundos
    global tarea

    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
    segundos += 1
    if segundos == 60:
        segundos = 0
        minutos += 1
    if minutos == 60:
        minutos = 0
        horas += 1
    tarea = rotulo_cronometro.after(1000, iniciar_cronometro)

    rotulo_cronometro.focus()
    boton_iniciar["state"] = "disabled"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "normal"

def parar_cronometro():
    global tarea

    rotulo_cronometro.after_cancel(tarea)
    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "normal"
    boton_parar["state"] = "disabled"

def resetear_cronometro():
    global horas, minutos, segundos
    horas, minutos,segundos = 0, 0, 0
    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "disabled"

##### FUNCIONES MENUS #####
def modo_oscuro():
    rotulo_fecha.config(background="black")
```

```
    rotulo_hora.config(background="black")
    rotulo_cronometro.config(background="black")
    menu_modos.entryconfig(0, label="Oscuro \u2713")
    menu_modos.entryconfig(1, label="Claro")

def modo_claro():
    rotulo_fecha.config(background="white")
    rotulo_hora.config(background="white")
    rotulo_cronometro.config(background="white")
    menu_modos.entryconfig(0, label="Oscuro")
    menu_modos.entryconfig(1, label="Claro \u2713")

def modo_pequenio():
    ventana.geometry("260x160+500+100")
    rotulo_fecha.config(font="arial 18 bold")
    rotulo_hora.config(font="arial 36")
    rotulo_cronometro.config(font="arial 30")
    boton_iniciar.config(font="arial 10")
    boton_parar.config(font="arial 10")
    boton_resetear.config(font="arial 10")
    menu_modos.entryconfig(3, label="Pequeño \u2713")
    menu_modos.entryconfig(4, label="Grande")

def modo_grande():
    ventana.geometry("900x450+300+100")
    rotulo_fecha.config(font="arial 60 bold")
    rotulo_hora.config(font="arial 120")
    rotulo_cronometro.config(font="arial 90")
    boton_iniciar.config(font="arial 30")
    boton_parar.config(font="arial 30")
    boton_resetear.config(font="arial 30")
    menu_modos.entryconfig(3, label="Pequeño")
    menu_modos.entryconfig(4, label="Grande \u2713")

##### CREACIÓN DEL CUADERNO #####
cuaderno = ttk.Notebook(ventana)
cuaderno.pack(expand = True, fill=tk.BOTH)

##### PAGINA 1 DEL CUADERNO #####
pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
cuaderno.add(pagina_1, text="  Fecha/Hora  ")

fecha = tk.StringVar()

rotulo_fecha = tk.Label(pagina_1,
    textvariable = fecha,
    font="arial 18 bold",
    bg="black", fg="green")
rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)
```

```
hora = tk.StringVar()

rotulo_hora = tk.Label(pagina_1,
    textvariable = hora,
    font="arial 36 bold",
    bg="black", fg="blue",
    anchor=tk.N)
rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)

actualizar_hora()
actualizar_fecha()

##### PAGINA 2 DEL CUADERNO #####
pagina_2 = tk.Frame(cuaderno)
cuaderno.add(pagina_2, text="  Cronómetro  ")

tiempo = tk.StringVar()
tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

rotulo_cronometro = tk.Label(pagina_2,
    textvariable = tiempo,
    font="arial 30",
    bg="black", fg="blue")
rotulo_cronometro.pack(expand=True, fill=tk.BOTH)

frame_botones = tk.Frame(pagina_2)

boton_iniciar = tk.Button(frame_botones,
    font="arial 10",
    text="Iniciar",
    width=6,
    bg="lightgrey", fg="red",
    command=iniciar_cronometro)
boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

boton_parar = tk.Button(frame_botones,
    font="arial 10",
    text="Parar",
    width=6,
    state="disabled",
    bg="lightgrey", fg="red",
    command=parar_cronometro)
boton_parar.pack(padx=10, pady=10, side=tk.LEFT)

boton_resetear = tk.Button(frame_botones,
    font="arial 10",
    text="Reset",
    width=6,
```

```
    state="disabled",
    bg ="lightgrey", fg="red",
    command=resetear_cronometro )
boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)

frame_botones.pack()

##### CREACIÓN DE LOS MENUS #####
# Creación de la barra de menús
barra_menus = tk.Menu(ventana)

# Cración de un menú
menu_modos = tk.Menu(barra_menus, tearoff=0)

# Creación de las opciones de un menú
menu_modos.add_command(label="Oscuro \u2713", command=modo_oscuro)
menu_modos.add_command(label="Claro", command=modo_claro)
menu_modos.add_separator()
menu_modos.add_command(label="Pequeño \u2713", command=modo_pequenio)
menu_modos.add_command(label="Grande", command=modo_grande)

# Colocamos el menú en la barra de menús
barra_menus.add_cascade(label="Modos", menu=menu_modos)

# Colocamos la barra de menús en la ventana
ventana.config(menu=barra_menus)
##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

Nota:

En los menús además de agregar .add_command se pueden agregar:

.add_checkbutton

.add_radiobutton

Para poder realizar menús más sofisticadas.

33.- Más menús

En este capítulo vamos a realizar un menú más avanzado.

Vamos a realizar un nuevo proyecto donde copiaremos el anterior.

Vamos a crear los menús:

```
201   ##### CREACIÓN DE LOS MENUS #####
202   opcion_tema=tk.IntVar()
203   opcion_tema.set(1)
204   opcion_fecha=tk.BooleanVar()
205   opcion_fecha.set(True)
206   opcion_hora=tk.BooleanVar()
207   opcion_hora.set(True)
208
209   # Creación de la barra de menús
210   barra_menus = tk.Menu(ventana)
211
212   menu_tamanio = tk.Menu(barra_menus, tearoff=0)
213   menu_tamanio.add_command(label="Aumentar", command=modo_aumentar)
214   menu_tamanio.add_command(label="Disminuir", state="disabled", command=modo_disminuir)
215   barra_menus.add_cascade(label="Tamaño", menu=menu_tamanio)
216
217  menu_tema = tk.Menu(barra_menus, tearoff=0)
218  menu_tema.add_radiobutton(label="Oscuro", value=1, variable=opcion_tema, command=modo_tema)
219  menu_tema.add_radiobutton(label="Claro", value=2, variable=opcion_tema, command=modo_tema)
220  barra_menus.add_cascade(label="Tema", menu=menu_tema)
221
222  menu_opciones = tk.Menu(barra_menus, tearoff=0)
223  menu_opciones.add_checkbutton(label="Fecha", variable=opcion_fecha, command=mostrar_fecha)
224  menu_opciones.add_checkbutton(label="Hora", variable=opcion_hora, command=mostrar_hora)
225  barra_menus.add_cascade(label="Opciones", menu=menu_opciones)
226
227  # Colocamos la barra de menús en la ventana
228  ventana.config(menu=barra_menus)
```

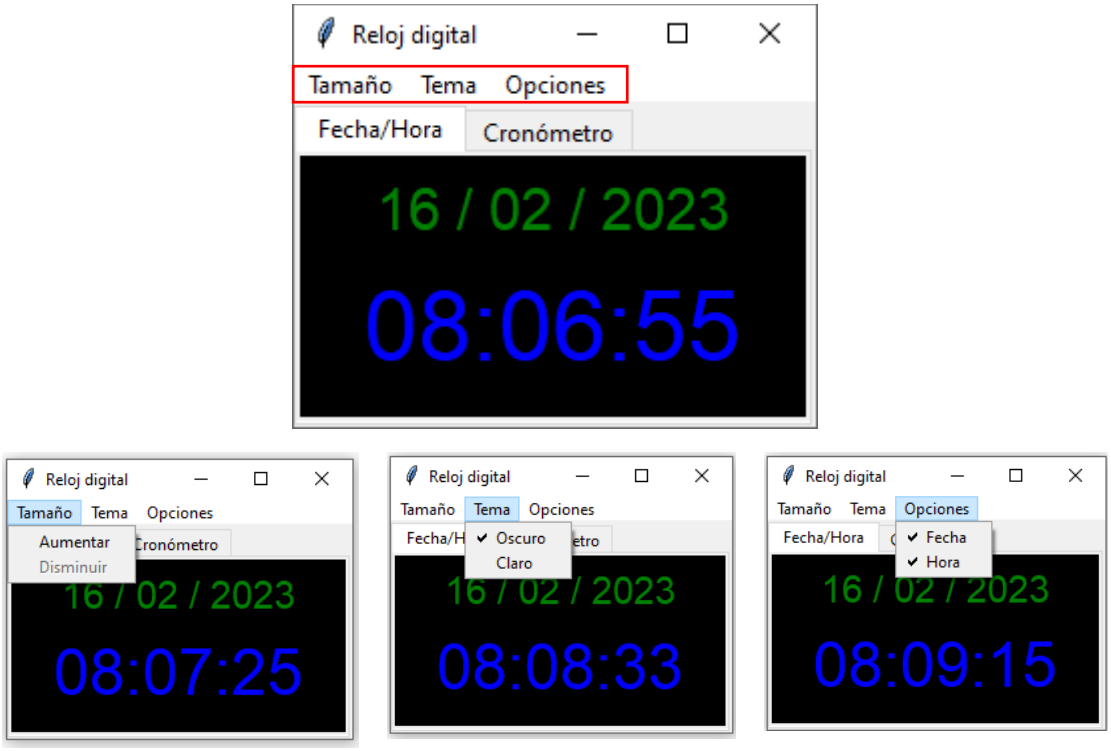
Estos menús tienen que remplazar a los anteriores.

Vamos a realizar las funciones que también se tienen que reemplazar por las anteriores.

```
60   ##### FUNCIONES MENUS #####
61   aumentos = 0
62   def modo_aumentar():
63       global aumentos
64       aumentos += 1
65       configurar_tamanio()
66
67   def modo_disminuir():
68       global aumentos
69       aumentos -= 1
70       configurar_tamanio()
71
72   def configurar_tamanio():
73       global aumentos
```

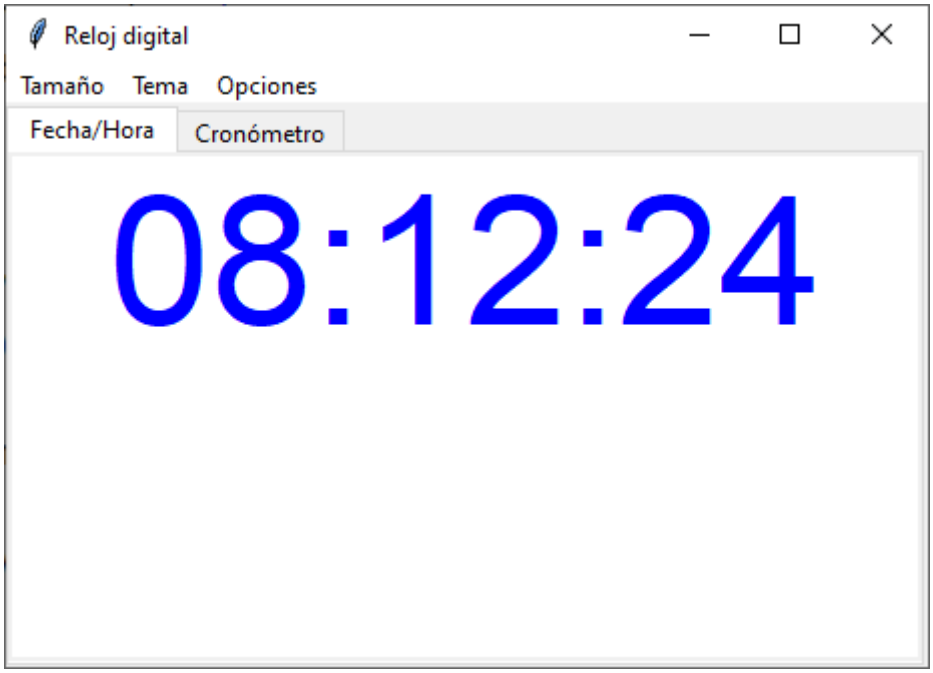
```
    if aumentos == 0:
        ventana.geometry("260x160+500+100")
        t_fecha, t_hora, t_crono, t_boton = 22, 36, 28, 10
        menu_tamanio.entryconfig(1, state="disabled")
    elif aumentos == 1:
        ventana.geometry("460x280+500+100")
        t_fecha, t_hora, t_crono, t_boton = 40, 68, 48, 18
        menu_tamanio.entryconfig(1, state="normal")
    elif aumentos == 2:
        ventana.geometry("580x340+500+100")
        t_fecha, t_hora, t_crono, t_boton = 52, 88, 58, 24
        menu_tamanio.entryconfig(0, state="normal")
    else:
        ventana.geometry("720x460+500+100")
        t_fecha, t_hora, t_crono, t_boton = 62, 108, 72, 36
        menu_tamanio.entryconfig(0, state="disabled")

    rotulo_fecha.config(font=("arial", t_fecha))
    rotulo_hora.config(font=("arial", t_hora))
    rotulo_cronometro.config(font=("arial", t_crono))
    boton_iniciar.config(font=("arial", t_boton))
    boton_parar.config(font=("arial", t_boton))
    boton_resetear.config(font=("arial", t_boton))

def modo_tema():
    if opcion_tema.get() == 1:
        rotulo_fecha.config(background="black")
        rotulo_hora.config(background="black")
        rotulo_cronometro.config(background="black")
    elif opcion_tema.get() == 2:
        rotulo_fecha.config(background="white")
        rotulo_hora.config(background="white")
        rotulo_cronometro.config(background="white")

def mostrar_fecha():
    if opcion_fecha.get():
        if opcion_hora.get():
            rotulo_hora.pack_forget()
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
            rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)
        else:
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
    else:
        rotulo_fecha.pack_forget()
```

```
119  def mostrar_hora():
120      if opcion_hora.get():
121          rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
122      else:
123          rotulo_hora.pack_forget()
```



Ahora tienes más combinaciones con el cronómetro.

Lo voy a hacer más grande, tema claro y que solo muestre la hora.



Adjunto todo el código para que lo puedas revisar.

```
import tkinter as tk
from tkinter import ttk
import time

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.title("Reloj digital")
ventana.geometry("260x160+900+100")

##### FUNCIONES RELOJ #####
def actualizar_hora():
    hora.set(time.strftime("%H:%M:%S"))
    rotulo_hora.after(500, actualizar_hora)
    rotulo_hora.focus()

def actualizar_fecha():
    fecha.set(time.strftime("%d / %m / %Y"))
    rotulo_fecha.after(500, actualizar_fecha)

##### FUNCIONES CRONOMETRO #####
horas, minutos,segundos = 0, 0, 0
tarea = None

def iniciar_cronometro():
    global horas, minutos, segundos
    global tarea

    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
    segundos += 1
    if segundos == 60:
        segundos = 0
        minutos += 1
    if minutos == 60:
        minutos = 0
        horas += 1
    tarea = rotulo_cronometro.after(1000, iniciar_cronometro)

    rotulo_cronometro.focus()
    boton_iniciar["state"] = "disabled"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "normal"

def parar_cronometro():
    global tarea

    rotulo_cronometro.after_cancel(tarea)
    boton_iniciar["state"] = "normal"
```

```
    boton_resetear["state"] = "normal"
    boton_parar["state"] = "disabled"

def resetear_cronometro():
    global horas, minutos, segundos
    horas, minutos,segundos = 0, 0, 0
    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "disabled"

##### FUNCIONES MENUS #####
aumentos = 0
def modo_aumentar():
    global aumentos
    aumentos += 1
    configurar_tamanio()

def modo_disminuir():
    global aumentos
    aumentos -= 1
    configurar_tamanio()

def configurar_tamanio():
    global aumentos
    if aumentos == 0:
        ventana.geometry("260x160+500+100")
        t_fecha, t_hora, t_crono, t_boton = 22, 36, 28, 10
        menu_tamanio.entryconfig(1, state="disabled")
    elif aumentos == 1:
        ventana.geometry("460x280+500+100")
        t_fecha, t_hora, t_crono, t_boton = 40, 68, 48, 18
        menu_tamanio.entryconfig(1, state="normal")
    elif aumentos == 2:
        ventana.geometry("580x340+500+100")
        t_fecha, t_hora, t_crono, t_boton = 52, 88, 58, 24
        menu_tamanio.entryconfig(0, state="normal")
    else:
        ventana.geometry("720x460+500+100")
        t_fecha, t_hora, t_crono, t_boton = 62, 108, 72, 36
        menu_tamanio.entryconfig(0, state="disabled")

    rotulo_fecha.config(font=("arial", t_fecha))
    rotulo_hora.config(font=("arial", t_hora))
    rotulo_cronometro.config(font=("arial", t_crono))
    boton_iniciar.config(font=("arial", t_boton))
    boton_parar.config(font=("arial", t_boton))
    boton_resetear.config(font=("arial", t_boton))
```

```
def modo_tema():
    if opcion_tema.get() == 1:
        rotulo_fecha.config(background="black")
        rotulo_hora.config(background="black")
        rotulo_cronometro.config(background="black")
    elif opcion_tema.get() == 2:
        rotulo_fecha.config(background="white")
        rotulo_hora.config(background="white")
        rotulo_cronometro.config(background="white")

def mostrar_fecha():
    if opcion_fecha.get():
        if opcion_hora.get():
            rotulo_hora.pack_forget()
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5,
ipady=5)
            rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=10,
ipady=10)
        else:
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5,
ipady=5)
    else:
        rotulo_fecha.pack_forget()

def mostrar_hora():
    if opcion_hora.get():
        rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
    else:
        rotulo_hora.pack_forget()

##### CREACIÓN DEL CUADERNO #####
cuaderno = ttk.Notebook(ventana)
cuaderno.pack(expand = True, fill=tk.BOTH)

##### PAGINA 1 DEL CUADERNO #####
pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
cuaderno.add(pagina_1, text="  Fecha/Hora  ")

fecha = tk.StringVar()

rotulo_fecha = tk.Label(pagina_1,
    textvariable = fecha,
    font="arial 18 bold",
    bg="black", fg="green")
rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)

hora = tk.StringVar()
```

```
rotulo_hora = tk.Label(pagina_1,
    textvariable = hora,
    font="arial 36 bold",
    bg="black", fg="blue",
    anchor=tk.N)
rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)

actualizar_hora()
actualizar_fecha()

##### PAGINA 2 DEL CUADERNO #####
pagina_2 = tk.Frame(cuaderno)
cuaderno.add(pagina_2, text="  Cronómetro  ")

tiempo = tk.StringVar()
tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

rotulo_cronometro = tk.Label(pagina_2,
    textvariable = tiempo,
    font="arial 30",
    bg="black", fg="blue")
rotulo_cronometro.pack(expand=True, fill=tk.BOTH)

frame_botones = tk.Frame(pagina_2)

boton_iniciar = tk.Button(frame_botones,
    font="arial 10",
    text="Iniciar",
    width=6,
    bg="lightgrey", fg="red",
    command=iniciar_cronometro)
boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

boton_parar = tk.Button(frame_botones,
    font="arial 10",
    text="Parar",
    width=6,
    state="disabled",
    bg="lightgrey", fg="red",
    command=parar_cronometro)
boton_parar.pack(padx=10, pady=10, side=tk.LEFT)

boton_resetear = tk.Button(frame_botones,
    font="arial 10",
    text="Reset",
    width=6,
    state="disabled",
    bg ="lightgrey", fg="red",
    command=resetear_cronometro )
```

```
boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)

frame_botones.pack()

##### CREACIÓN DE LOS MENUS #####
opcion_tema=tk.IntVar()
opcion_tema.set(1)
opcion_fecha=tk.BooleanVar()
opcion_fecha.set(True)
opcion_hora=tk.BooleanVar()
opcion_hora.set(True)

# Creación de la barra de menús
barra_menus = tk.Menu(ventana)

menu_tamanio = tk.Menu(barra_menus, tearoff=0)
menu_tamanio.add_command(label="Aumentar", command=modo_aumentar)
menu_tamanio.add_command(label="Disminuir", state="disabled",
command=modo_disminuir)
barra_menus.add_cascade(label="Tamaño", menu=menu_tamanio)

menu_tema = tk.Menu(barra_menus, tearoff=0)
menu_tema.add_radiobutton(label="Oscuro", value=1, variable=opcion_tema,
command=modo_tema)
menu_tema.add_radiobutton(label="Claro", value=2, variable=opcion_tema,
command=modo_tema)
barra_menus.add_cascade(label="Tema", menu=menu_tema)

menu_opciones = tk.Menu(barra_menus, tearoff=0)
menu_opciones.add_checkbutton(label="Fecha", variable=opcion_fecha,
command=mostrar_fecha)
menu_opciones.add_checkbutton(label="Hora", variable=opcion_hora,
command=mostrar_hora)
barra_menus.add_cascade(label="Opciones", menu=menu_opciones)

# Colocamos la barra de menús en la ventana
ventana.config(menu=barra_menus)
##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

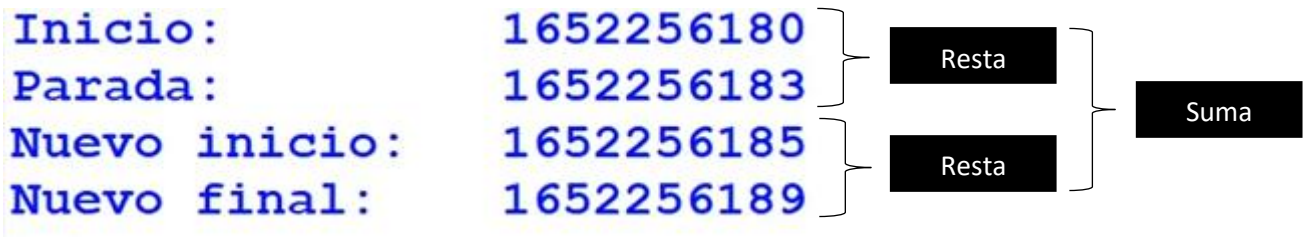
34.- Cronómetro mejorado

Vamos a realizar una mejora con respecto a lo que es el cronómetro.

Para que a la hora de cronometrar sea más preciso vamos a coger el tiempo desde el sistema.

```
import time

inicio = int(time.time())
print(f"Inicio:        {inicio:12d}")
time.sleep(3)
parada = int(time.time())
print(f"Parada:        {parada:12d}")
time.sleep(2)
nuevo_inicio = int(time.time())
print(f"Nuevo inicio: {nuevo_inicio:12d}")
time.sleep(4)
nuevo_final = int(time.time())
print(f"Nuevo final:   {nuevo_final:12d}")
```

Teniendo en cuenta que podemos para el cronometro y a continuación reanudarlo, vamos a ver como lo planteamos:




```
>>> 1652256189 - 1652256185 + (1652256183 - 1652256180)
7
```

Nos da 7 segundos.

Podemos hacer de otra forma que será mucho más fácil.

```
>>> 1652256189 - (1652256185 - (1652256183 - 1652256180))
7
```

El resultado será el mismo.

```
20    ##### FUNCIONES CRONOMETRO MEJORADO #####
21    inicio = 0
22    duracion = 0
23    tarea = None
```

```
24
25  def activar_cronometro():
26      global inicio
27      global duracion
28      if duracion == 0:
29          inicio = time.time()
30      else:
31          inicio = time.time() - duracion
32
33      rotulo_cronometro.after(300, actualizar_cronometro)
34
35      rotulo_cronometro.focus()
36      boton_iniciar["state"] = "disabled"
37      boton_resetear["state"] = "disabled"
38      boton_parar["state"] = "normal"
39
40  def actualizar_cronometro():
41      global inicio
42      global tarea
43      global duracion
44      duracion = time.time() - inicio
45      horas = int(duracion // 60 // 60)
46      minutos = int(duracion // 60 % 60)
47      segundos = int(duracion % 60)
48
49      tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
50      tarea = rotulo_cronometro.after(300, actualizar_cronometro)
51
52  def resetear_cronometro():
53      global horas, minutos, segundos
54      horas, minutos,segundos = 0, 0, 0
55      tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
56
57      boton_iniciar["state"] = "normal"
58      boton_resetear["state"] = "disabled"
59      boton_parar["state"] = "disabled"
60
61  def parar_cronometro():
62      global tarea
63      rotulo_cronometro.after_cancel(tarea)
64      boton_iniciar["state"] = "normal"
65      boton_resetear["state"] = "normal"
66      boton_parar["state"] = "disabled"
67
```

```
68  def resetear_cronometro():
69      global inicio
70      global duracion
71      inicio = 0
72      duracion = 0
73
74      tiempo.set(f"00:00:00")
75      boton_iniciar["state"] = "normal"
76      boton_resetear["state"] = "disabled"
77      boton_parar["state"] = "disabled"
```

Eliminamos todas las funciones del cronometro para sustituirlas por estas nuevas.

```
176  tiempo = tk.StringVar()
177  tiempo.set(f"00:00:00")
```

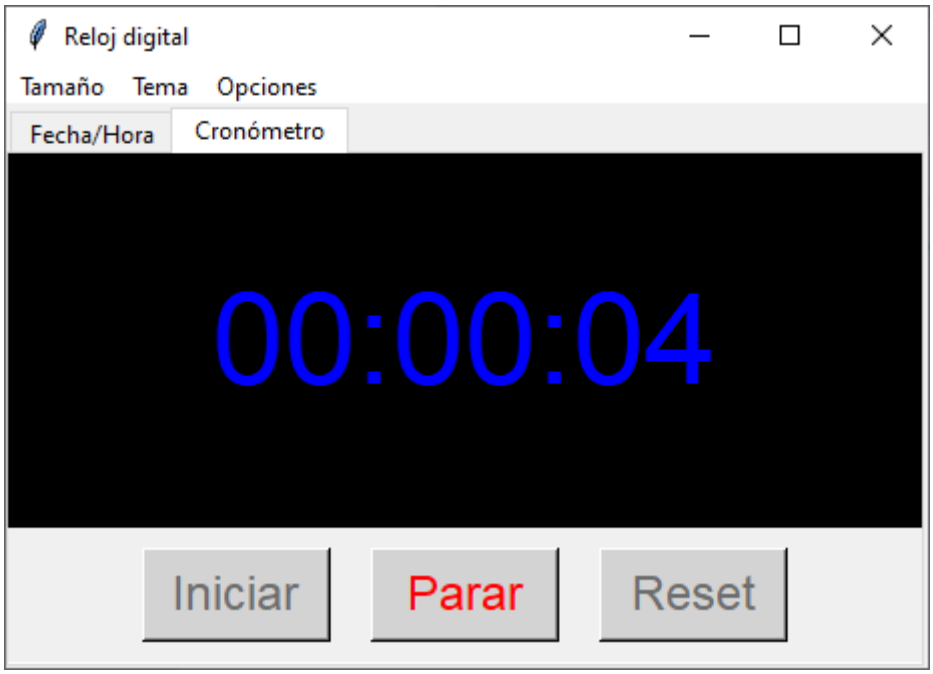
Cambiamos el valor de la variable tiempo de tipo tkinter.

Los botones llamarán a sus correspondientes funciones:

```
187  boton_iniciar = tk.Button(frame_botones,
188      font="arial 10",
189      text="Iniciar",
190      width=6,
191      bg="lightgrey", fg="red",
192      command=activar_cronometro)
193  boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

195   boton_parar = tk.Button(frame_botones,
196       font="arial 10",
197       text="Parar",
198       width=6,
199       state="disabled",
200       bg="lightgrey", fg="red",
201       command=parar_cronometro)
202   boton_parar.pack(padx=10, pady=10, side=tk.LEFT)

204  boton_resetear = tk.Button(frame_botones,
205      font="arial 10",
206      text="Reset",
207      width=6,
208      state="disabled",
```

```
209         bg ="lightgrey", fg="red",
210         command=resetear_cronometro )
211     boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)
```

A partir de ahora el cronómetro será más preciso.



Te adjunto el codigo completo para posibles consultas.

```
import tkinter as tk
from tkinter import ttk
import time

##### VENTANA PRINCIPAL #####
ventana = tk.Tk()
ventana.title("Reloj digital")
ventana.geometry("260x160+900+100")

##### FUNCIONES RELOJ #####
def actualizar_hora():
    horas.set(time.strftime("%H:%M:%S"))
    rotulo_hora.after(500, actualizar_hora)
    rotulo_hora.focus()

def actualizar_fecha():
    fecha.set(time.strftime("%d / %m / %Y"))
    rotulo_fecha.after(500, actualizar_fecha)

##### FUNCIONES CRONOMETRO MEJORADO #####
inicio = 0
duracion = 0
tarea = None
```

```
def activar_cronometro():
    global inicio
    global duracion
    if duracion == 0:
        inicio = time.time()
    else:
        inicio = time.time() - duracion

    rotulo_cronometro.after(300, actualizar_cronometro)

    rotulo_cronometro.focus()
    boton_iniciar["state"] = "disabled"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "normal"

def actualizar_cronometro():
    global inicio
    global tarea
    global duracion
    duracion = time.time() - inicio
    horas = int(duracion // 60 // 60)
    minutos = int(duracion // 60 % 60)
    segundos = int(duracion % 60)

    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")
    tarea = rotulo_cronometro.after(300, actualizar_cronometro)

def resetear_cronometro():
    global horas, minutos, segundos
    horas, minutos,segundos = 0, 0, 0
    tiempo.set(f"{horas:02d}:{minutos:02d}:{segundos:02d}")

    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "disabled"

def parar_cronometro():
    global tarea
    rotulo_cronometro.after_cancel(tarea)
    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "normal"
    boton_parar["state"] = "disabled"

def resetear_cronometro():
    global inicio
    global duracion
    inicio = 0
    duracion = 0
```

```
    tiempo.set(f"00:00:00")
    boton_iniciar["state"] = "normal"
    boton_resetear["state"] = "disabled"
    boton_parar["state"] = "disabled"

##### FUNCIONES MENUS #####
aumentos = 0
def modo_aumentar():
    global aumentos
    aumentos += 1
    configurar_tamanio()

def modo_disminuir():
    global aumentos
    aumentos -= 1
    configurar_tamanio()

def configurar_tamanio():
    global aumentos
    if aumentos == 0:
        ventana.geometry("260x160+500+100")
        t_fecha, t_hora, t_crono, t_boton = 22, 36, 28, 10
        menu_tamanio.entryconfig(1, state="disabled")
    elif aumentos == 1:
        ventana.geometry("460x280+500+100")
        t_fecha, t_hora, t_crono, t_boton = 40, 68, 48, 18
        menu_tamanio.entryconfig(1, state="normal")
    elif aumentos == 2:
        ventana.geometry("580x340+500+100")
        t_fecha, t_hora, t_crono, t_boton = 52, 88, 58, 24
        menu_tamanio.entryconfig(0, state="normal")
    else:
        ventana.geometry("720x460+500+100")
        t_fecha, t_hora, t_crono, t_boton = 62, 108, 72, 36
        menu_tamanio.entryconfig(0, state="disabled")

    rotulo_fecha.config(font=("arial", t_fecha))
    rotulo_hora.config(font=("arial", t_hora))
    rotulo_cronometro.config(font=("arial", t_crono))
    boton_iniciar.config(font=("arial", t_boton))
    boton_parar.config(font=("arial", t_boton))
    boton_resetear.config(font=("arial", t_boton))

def modo_tema():
    if opcion_tema.get() == 1:
        rotulo_fecha.config(background="black")
        rotulo_hora.config(background="black")
        rotulo_cronometro.config(background="black")
```

```
    elif opcion_tema.get() == 2:
        rotulo_fecha.config(background="white")
        rotulo_hora.config(background="white")
        rotulo_cronometro.config(background="white")

def mostrar_fecha():
    if opcion_fecha.get():
        if opcion_hora.get():
            rotulo_hora.pack_forget()
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5,
ipady=5)
            rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=10,
ipady=10)
        else:
            rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=5,
ipady=5)
    else:
        rotulo_fecha.pack_forget()

def mostrar_hora():
    if opcion_hora.get():
        rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
    else:
        rotulo_hora.pack_forget()

##### CREACIÓN DEL CUADERNO #####
cuaderno = ttk.Notebook(ventana)
cuaderno.pack(expand = True, fill=tk.BOTH)

##### PAGINA 1 DEL CUADERNO #####
pagina_1 = tk.Frame(cuaderno, padx=2, pady=2)
cuaderno.add(pagina_1, text="  Fecha/Hora  ")

fecha = tk.StringVar()

rotulo_fecha = tk.Label(pagina_1,
    textvariable = fecha,
    font="arial 18 bold",
    bg="black", fg="green")
rotulo_fecha.pack(expand=True, fill=tk.BOTH, ipadx=10, ipady=10)

horas = tk.StringVar()

rotulo_hora = tk.Label(pagina_1,
    textvariable = horas,
    font="arial 36 bold",
    bg="black", fg="blue",
    anchor=tk.N)
rotulo_hora.pack(expand=True, fill=tk.BOTH, ipadx=5, ipady=5)
```

```
actualizar_hora()
actualizar_fecha()

##### PAGINA 2 DEL CUADERNO #####
pagina_2 = tk.Frame(cuaderno)
cuaderno.add(pagina_2, text="  Cronómetro  ")

tiempo = tk.StringVar()
tiempo.set(f"00:00:00")

rotulo_cronometro = tk.Label(pagina_2,
    textvariable = tiempo,
    font="arial 30",
    bg="black", fg="blue")
rotulo_cronometro.pack(expand=True, fill=tk.BOTH)

frame_botones = tk.Frame(pagina_2)

boton_iniciar = tk.Button(frame_botones,
    font="arial 10",
    text="Iniciar",
    width=6,
    bg="lightgrey", fg="red",
    command=activar_cronometro)
boton_iniciar.pack(padx=10, pady=10, side=tk.LEFT)

boton_parar = tk.Button(frame_botones,
    font="arial 10",
    text="Parar",
    width=6,
    state="disabled",
    bg="lightgrey", fg="red",
    command=parar_cronometro)
boton_parar.pack(padx=10, pady=10, side=tk.LEFT)

boton_resetear = tk.Button(frame_botones,
    font="arial 10",
    text="Reset",
    width=6,
    state="disabled",
    bg ="lightgrey", fg="red",
    command=resetear_cronometro )
boton_resetear.pack(padx=10, pady=10, side=tk.LEFT)

frame_botones.pack()

##### CREACIÓN DE LOS MENUS #####
opcion_tema=tk.IntVar()
```

```
opcion_tema.set(1)
opcion_fecha=tk.BooleanVar()
opcion_fecha.set(True)
opcion_hora=tk.BooleanVar()
opcion_hora.set(True)

# Creación de la barra de menús
barra_menus = tk.Menu(ventana)

menu_tamanio = tk.Menu(barra_menus, tearoff=0)
menu_tamanio.add_command(label="Aumentar", command=modo_aumentar)
menu_tamanio.add_command(label="Disminuir", state="disabled",
command=modo_disminuir)
barra_menus.add_cascade(label="Tamaño", menu=menu_tamanio)

menu_tema = tk.Menu(barra_menus, tearoff=0)
menu_tema.add_radiobutton(label="Oscuro", value=1, variable=opcion_tema,
command=modo_tema)
menu_tema.add_radiobutton(label="Claro", value=2, variable=opcion_tema,
command=modo_tema)
barra_menus.add_cascade(label="Tema", menu=menu_tema)

menu_opciones = tk.Menu(barra_menus, tearoff=0)
menu_opciones.add_checkbutton(label="Fecha", variable=opcion_fecha,
command=mostrar_fecha)
menu_opciones.add_checkbutton(label="Hora", variable=opcion_hora,
command=mostrar_hora)
barra_menus.add_cascade(label="Opciones", menu=menu_opciones)

# Colocamos la barra de menús en la ventana
ventana.config(menu=barra_menus)
##### BUCLE PRINCIPAL #####
ventana.mainloop()
```

Contenido

1. Introducción 1
2.- Primera ventana en tkinter 5
3.- Rótulos o etiquetas con el widget Label 10
4.- Botones en tkinter con el widget Button 17
5.- Dando funcionalidad a los botones de tkinter con la opción command 20
6.- Método pack com0 geometry manager 24
7.- Widget Frame en tkinter par conversor de temperaturas 33
8.- Caja de texto con el widget Entry para la entrada de datos. 37
9.- Método get para las cajas de texto o widgets Entry 40
10.- Deshabilitar widgets 42
11.- Variables StringVar() en tkinter 48
12.- Widget Radiobutton en tkinter 52
13.- Funcionalidad a botones de opciones o radio botones 57
14.- Opciones command con los radio botones 59
15.- Widget Checkbutton 67
16.- El módulo ttk de tkinter 69
17.- Widget Combobox 70
18.- Método bind con teclas 76
19.- Método bind con eventos del ratón 81
20.- Método bind con el widget Combobox 87
21.- El método grid de tkinter 94
22.- Opción sticky para el método grid 101
23.- Método grid columnconfigure 107
24.- Calculadora con el método grid. 109
25.- Argumentos y funciones mediante funciones lambda 116
26.- Funcionalidad de la calculadora de tkinter 119
27.- Convertir un archivo .py a un archivo .exe 130
28.- Método after de los widgets en tkinter 134
29.- Método after_cancel 137
30.- Reto 30 – Widget Notebook 138
31.- Aplicación de reloj digital 144
32.- Widget Menu 151
33.- Más menús 158
34.- Cronómetro mejorado 166