EBook Gratis

APRENDIZAJE tkinter

Free unaffiliated eBook created from
Stack Overflow contributors.

#tkinter

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: tkinter

It is an unofficial and free tkinter ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official tkinter.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con tkinter

Observaciones

Tkinter (" **Tk Inter** face") es un paquete multiplataforma estándar de python para crear interfaces gráficas de usuario (GUI). Proporciona acceso a un intérprete de Tcl subyacente con el kit de herramientas Tk, que en sí mismo es una biblioteca de interfaz de usuario gráfica multiplataforma y multiplataforma.

Tkinter no es la única biblioteca GUI para python, pero es la que viene de serie. Las bibliotecas de GUI adicionales que se pueden usar con python incluyen wxPython , PyQt y kivy .

La mayor fortaleza de Tkinter es su ubicuidad y simplicidad. Funciona de forma inmediata en la mayoría de las plataformas (Linux, OSX, Windows) y se completa con una amplia gama de widgets necesarios para las tareas más comunes (botones, etiquetas, lienzos de dibujo, texto de varias líneas, etc.).

Como herramienta de aprendizaje, tkinter tiene algunas características que son únicas entre los kits de herramientas GUI, como las fuentes con nombre, las etiquetas de enlace y el rastreo de variables.

Diferencias entre python 2 y 3

Tkinter se mantiene prácticamente sin cambios entre python 2 y python 3, con la diferencia principal de que se cambió el nombre del paquete tkinter y de los módulos.

Importando en python 2.x

En Python 2.x, el paquete tkinter se llama `Tkinter` , y los paquetes relacionados tienen sus propios nombres. Por ejemplo, lo siguiente muestra un conjunto típico de instrucciones de importación para python 2.x:

```
import Tkinter as tk
import tkFileDialog as filedialog
import ttk
```

Importando en Python 3.x

Aunque la funcionalidad no cambió mucho entre python 2 y 3, los nombres de todos los módulos tkinter han cambiado. El siguiente es un conjunto típico de instrucciones de importación para Python 3.x:

```
import tkinter as tk
from tkinter import filedialog
```

```
from tkinter import ttk
```

Otras lecturas

- Preguntas Tkinter en Stackoverflow
- Documentación oficial de Python 3 tkinter
- Documentación oficial de Python 2 tkinter
- Tkdocs.com - multiplataforma tk documentación
- Introducción de Effbot a tkinter
- Guía de referencia Tkinter, New Mexico Tech

Versiones

Tcl

Versión	Fecha de lanzamiento
8.6	2016-07-27
8.5	2016-02-12
8.4	2013-06-01
8.3	2002-10-18
8.2	1999-12-16
8.1	1999-05-26
8.0	1999-03-09

Pitón

Versión	Fecha de lanzamiento
3.6	2016-12-23
3.5	2015-09-13
3.4	2014-03-17
3.3	2012-09-29
3.2	2011-02-20

Versión	Fecha de lanzamiento
3.1	2009-06-26
3.0	2008-12-03
2.7	2010-07-03
2.6	2008-10-02
2.5	2006-09-19
2.4	2004-11-30
2.3	2003-07-29
2.2	2001-12-21
2.1	2001-04-15
2.0	2000-10-16

Examples

Instalación o configuración

Tkinter viene preinstalado con los binarios del instalador de Python para Mac OS X y la plataforma Windows. Por lo tanto, si instala Python desde los archivos binarios oficiales para Mac OS X o la plataforma Windows, puede utilizar Tkinter.

Para las versiones Debian de Linux, debe instalarlo manualmente utilizando los siguientes comandos.

Para Python 3

```
sudo apt-get install python3-tk
```

Para Python 2.7

```
sudo apt-get install python-tk
```

Las distribuciones de Linux con yum installer pueden instalar el módulo tkinter usando el comando:

```
yum instalar tkinter
```

Verificando instalación

Para verificar si ha instalado Tkinter correctamente, abra su consola de Python y escriba el siguiente comando:

```
import tkinter as tk # for Python 3 version
```

o

```
import Tkinter as tk # for Python 2.x version
```

Ha instalado Tkinter correctamente, si el comando anterior se ejecuta sin un error.

Para verificar la versión de Tkinter, escriba los siguientes comandos en su REPL de Python:

Para python 3.x

```
import tkinter as tk
tk._test()
```

Para python 2.x

```
import Tkinter as tk
tk._test()
```

Nota: Importar `Tkinter as tk` no es obligatorio, pero es una buena práctica ya que ayuda a mantener la coherencia entre las versiones.

¡Hola Mundo! (mínimo)

Probemos nuestro conocimiento básico de tkinter creando el clásico "¡Hola mundo!" programa.

Primero, debemos importar tkinter, esto variará según la versión (consulte la sección de comentarios sobre "Diferencias entre Python 2 y 3")

En Python 3, el módulo `tkinter` tiene una t minúscula:

```
import tkinter as tk
```

En Python 2, el módulo `Tkinter` tiene una T mayúscula:

```
import Tkinter as tk
```

Usar `as tk` no es estrictamente necesario, pero lo usaremos para que el resto de este ejemplo funcione de la misma manera para ambas versiones.

Ahora que hemos importado el módulo tkinter, podemos crear la raíz de nuestra aplicación usando la clase `Tk` :

```
root = tk.Tk()
```

Esto actuará como la ventana para nuestra aplicación. (Tenga en cuenta que las ventanas *adicionales* deben ser instancias de `Toplevel` lugar)

Ahora que tenemos una ventana, vamos a agregarle texto con una `Label`

```
label = tk.Label(root, text="Hello World!") # Create a text label
label.pack(padx=20, pady=20) # Pack it into the window
```

Una vez que la aplicación esté lista, podemos iniciarla (ingresar al *bucle del* evento *principal*) con el método `mainloop`

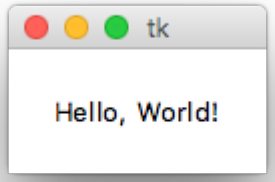
```
root.mainloop()
```

Esto abrirá y ejecutará la aplicación hasta que se detenga al cerrar la ventana o al llamar a las funciones que salen de las devoluciones de llamada (que se `root.destroy()` más adelante) como `root.destroy()` .

Poniendolo todo junto:

```
import tkinter as tk # Python 3.x Version
#import Tkinter as tk # Python 2.x Version

root = tk.Tk()

label = tk.Label(root, text="Hello World!") # Create a text label
label.pack(padx=20, pady=20) # Pack it into the window

root.mainloop()
```

Y algo como esto debería aparecer:



¡Hola Mundo! (modular, orientado a objetos)

```
import tkinter as tk

class HelloWorld(tk.Frame):
    def __init__(self, parent):
        super(HelloWorld, self).__init__(parent)

        self.label = tk.Label(self, text="Hello, World!")
        self.label.pack(padx=20, pady=20)

if __name__ == "__main__":
    root = tk.Tk()
```

```
    main = HelloWorld(root)
    main.pack(fill="both", expand=True)

    root.mainloop()
```

Nota: es posible heredar de casi cualquier widget tkinter, incluida la ventana raíz. Heredar de `tkinter.Frame` es, al menos, posiblemente el más flexible, ya que admite interfaces de documentos múltiples (MDI), interfaces de documentos únicos (SDI), aplicaciones de página única y aplicaciones de página múltiple.

Lea Empezando con tkinter en línea: https://riptutorial.com/es/tkinter/topic/987/empezando-con-tkinter

Capítulo 2: Agregar imágenes a la etiqueta / botón

Introducción

Esto muestra el uso correcto de las imágenes y cómo visualizarlas correctamente.

Examples

Formatos de archivo soportados por Tkinter

Tkinter admite archivos .ppm de PIL (Python Imaging Library), .JPG, .PNG y .GIF.

Para importar una imagen, primero necesita crear una referencia como esta:

```
Image = PhotoImage(filename = [Your Image here])
```

Ahora, podemos agregar esta imagen a los botones y etiquetas como usando la devolución de llamada "img":

```
 Lbl = Label (width=490, img=image)
```

Uso de formatos .GIF.

Para mostrar un gif, debes mostrarlo fotograma a fotograma como una animación.

Un gif animado consiste en una serie de cuadros en un solo archivo. Tk carga el primer fotograma, pero puede especificar diferentes fotogramas pasando un parámetro de índice al crear la imagen. Por ejemplo:

```
frame2 = PhotoImage(file=imagefilename, format="gif -index 2")
```

Si carga todos los marcos en PhotoImages separados y luego usa eventos de temporizador para cambiar el marco que se muestra (label.configure (image = nextframe)). El retraso en el temporizador le permite controlar la velocidad de la animación. No hay nada que le proporcione el número de cuadros en la imagen que no sea la creación de un cuadro una vez que supera el recuento de cuadros.

Lea Agregar imágenes a la etiqueta / botón en línea:
https://riptutorial.com/es/tkinter/topic/9746/agregar-imagenes-a-la-etiqueta---boton

Capítulo 3: El widget de entrada Tkinter

Sintaxis

- entrada = tk.Entry (*padre* , ** *kwargs*)
- entry.get ()
- entry.insert (índice, "valor")
- entry.delete (start_index, end_index)
- entry.bind (event, callback)

Parámetros

Parámetro	Descripción
padre	Los widgets tkinter existen en una jerarquía. A excepción de la ventana raíz, todos los widgets tienen un padre. Algunos tutoriales en línea llaman a esto "maestro". Cuando el widget se agregue a la pantalla con `pack` , `place` o `grid` , aparecerá dentro de este widget principal
anchura	El ancho especifica el ancho *deseado* del widget basado en un ancho de caracteres promedio. Para fuentes de ancho variable, esto se basa en el ancho del carácter cero (`0`). El valor predeterminado es 20. Tenga en cuenta que el ancho real podría ser mayor o menor según cómo se agregue a la pantalla.

Observaciones

Estos ejemplos asumen que tkinter se ha importado con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Examples

Creación de un widget de entrada y configuración de un valor predeterminado

```
entry = tk.Entry(parent, width=10)
entry.insert(0, "Hello, World!")
```

Obtener el valor de un widget de entrada

El valor de un widget de entrada se puede obtener con el método `get` del widget:

```
name_entry = tk.Entry(parent)
...
name = name_entry.get()
```

Opcionalmente, puede asociar una instancia de `StringVar` y recuperar el valor de `StringVar` lugar de hacerlo desde el widget:

```
name_var = tk.StringVar()
name_entry = tk.Entry(parent, textvariable=name_var)
...
name = name_var.get()
```

Añadiendo validación a un widget de entrada

Para restringir los caracteres que se pueden escribir en un widget de entrada, solo números, por ejemplo, se puede agregar un comando de validación a la entrada. Un comando de validación es una función que devuelve `True` si se acepta el cambio, `False` contrario. Esta función se llamará cada vez que se modifique el contenido de la entrada. Se pueden pasar varios argumentos a esta función, como el tipo de cambio (inserción, eliminación), el texto insertado, ...

```
def only_numbers(char):
    return char.isdigit()

validation = parent.register(only_numbers)
entry = Entry(parent, validate="key", validatecommand=(validation, '%S'))
```

La opción de `validate` determina el tipo de evento que desencadena la validación, aquí, es cualquier pulsación de tecla en la entrada. El `'%S'` en la opción validatecommand significa que el carácter insertado o eliminado se pasa en argumento a la función `only_numbers` . La lista completa de posibilidades se puede encontrar aquí .

Obtención de int desde el widget de entrada

Al usar el método .get (), lo que sea que esté en el widget de entrada se convertirá en una cadena. Por ejemplo, independientemente del tipo de entrada (puede ser un número o una oración), el resultado resultante será una cadena. Si el usuario escribe 4, la salida será "4" como en una cadena. Para obtener un int desde un widget de entrada, primero llame al método .get ().

```
What_User_Wrote = Entry.get()
```

Ahora convertimos esa cadena en un int así:

```
Convert_To_Int = int(What_User_Wrote)
```

Del mismo modo, si quieres ahorrar tiempo puedes simplemente hacer:

```
Convert_To_Int = int(Entry.get())
```

Puede utilizar el método anterior si no desea convertir str a int.

Lea El widget de entrada Tkinter en línea: https://riptutorial.com/es/tkinter/topic/4868/el-widget-de-entrada-tkinter

Capítulo 4: El widget Tkinter Radiobutton

Sintaxis

- radiobutton = tk.Radiobutton (padre, ** kwargs)

Parámetros

Parámetro	Descripción
padre	Los widgets tkinter existen en una jerarquía. A excepción de la ventana raíz, todos los widgets tienen un padre. Algunos tutoriales en línea llaman a esto "maestro". Cuando el widget se agregue a la pantalla con el paquete, el lugar o la cuadrícula, aparecerá dentro de este widget principal.
mando	Función llamada cada vez que el usuario cambia el estado del botón de radio
indicador de	1 o Verdadero para los botones de opción, 0 o Falso para los cuadros de botones
texto	Texto para mostrar junto al botón de radio.
valor	Cuando se selecciona el botón de radio, la variable de control asociada se establece en valor.
variable	Variable de control que el botón de radio comparte con el otro botón de radio del grupo.

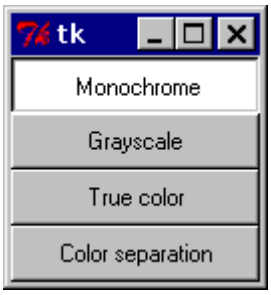
Observaciones

Estos ejemplos asumen que tkinter se ha importado con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Referencia:

Para convertir el ejemplo anterior en un "cuadro de botones" en lugar de un conjunto de botones de radio, establezca la opción de indicador en 0. En este caso, no hay un

indicador de botón de radio por separado, y el botón seleccionado se dibuja como SUNKEN en lugar de AUMENTADO:



- Effbot

Examples

Aquí hay un ejemplo de cómo convertir los botones de opción en los cuadros de botones:

```
import tkinter as tk
root = tk.Tk()

rbvar = StringVar()
rbvar.set(" ")

rb1 = tk.Radiobutton(root, text="Option 1", variable=rbvar, value='a', indicatoron=0)
rb1.pack()

rb2 = tk.Radiobutton(root, text="Option 2", variable=rbvar, value='b', indicatoron=0)
rb2.pack()
```

Crear un grupo de botones de radio.

Dicho grupo está formado por botones de radio que comparten una variable de control de modo que no se puede seleccionar más de uno.

```
# control variable
var = tk.IntVar(parent, 0)

# group of radiobuttons
for i in range(1,4):
    tk.Radiobutton(parent, text='Choice %i' % i, value=i, variable=var).pack()

tk.Button(parent, text='Print choice', command=lambda: print(var.get())).pack()
```

Lea El widget Tkinter Radiobutton en línea: https://riptutorial.com/es/tkinter/topic/6338/el-widget-tkinter-radiobutton

Capítulo 5: Personaliza los estilos ttk.

Introducción

El estilo de los nuevos widgets ttk es uno de los aspectos más poderosos de ttk. Además del hecho de que es una forma de trabajar completamente diferente a la del paquete tk tradicional, permite realizar un gran grado de personalización en tus widgets.

Examples

Personaliza una vista de árbol

Tomando Treeview: Ejemplo básico , se puede mostrar cómo personalizar una vista de árbol básica.

En este caso, creamos un estilo "mystyle.Treeview" con el siguiente código (vea los comentarios para comprender lo que hace cada línea):

```
style = ttk.Style()
style.configure("mystyle.Treeview", highlightthickness=0, bd=0, font=('Calibri', 11)) # Modify
the font of the body
style.configure("mystyle.Treeview.Heading", font=('Calibri', 13,'bold')) # Modify the font of
the headings
style.layout("mystyle.Treeview", [('mystyle.Treeview.treearea', {'sticky': 'nswe'})]) # Remove
the borders
```

Entonces, el widget se crea dando el estilo anterior:

```
tree=ttk.Treeview(master,style="mystyle.Treeview")
```

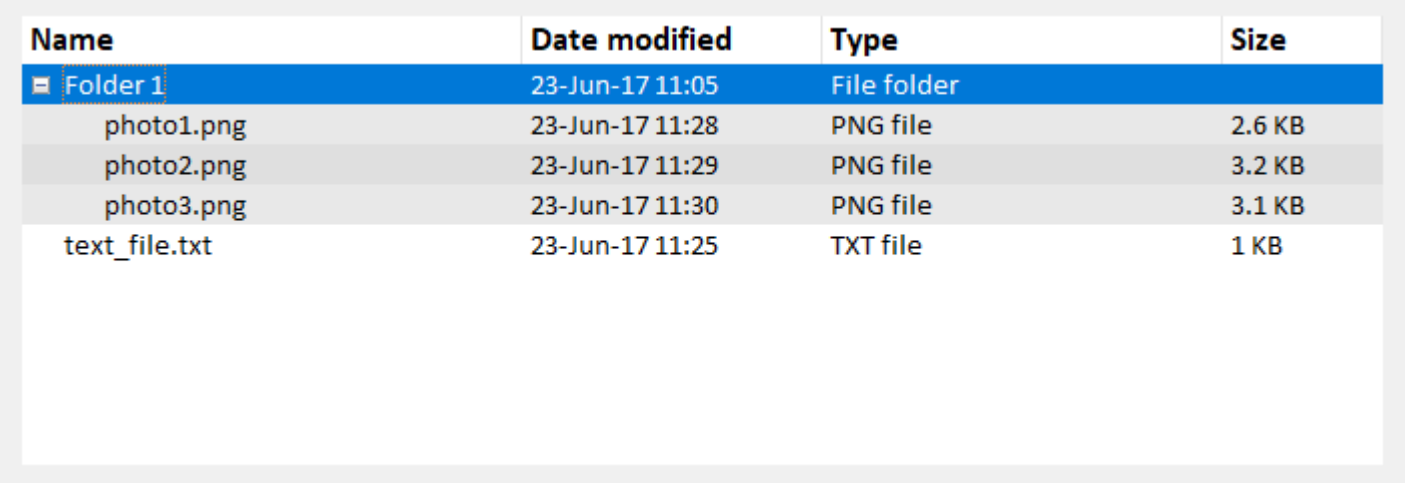
Si desea tener un formato diferente dependiendo de las filas, puede hacer uso de `tags` :

```
tree.insert(folder1, "end", "", text="photo1.png", values=("23-Jun-17 11:28","PNG file","2.6
KB"),tags = ('odd',))
tree.insert(folder1, "end", "", text="photo2.png", values=("23-Jun-17 11:29","PNG file","3.2
KB"),tags = ('even',))
tree.insert(folder1, "end", "", text="photo3.png", values=("23-Jun-17 11:30","PNG file","3.1
KB"),tags = ('odd',))
```

Luego, por ejemplo, se puede asociar un color de fondo a las etiquetas:

```
tree.tag_configure('odd', background='#E8E8E8')
tree.tag_configure('even', background='#DFDFDF')
```

El resultado es una vista de árbol con fuentes modificadas tanto en el cuerpo como en los encabezados, sin bordes y colores diferentes para las filas:



Nota: Para generar la imagen anterior, debe agregar / cambiar las líneas de código mencionadas anteriormente en el ejemplo Treeview: Basic ejemplo .

Lea Personaliza los estilos ttk. en línea: https://riptutorial.com/es/tkinter/topic/10624/personaliza-los-estilos-ttk-

Capítulo 6: Retrasando una función

Sintaxis

- widget.after (delay_ms, callback, * args)

Parámetros

Parámetro	Descripción
delay_ms	Tiempo (milisegundos) que se retrasa la llamada a la función de `callback`
llamar de vuelta	Función que se llama después del `delay_ms` dado. Si este parámetro no se proporciona, `.after` actúa de manera similar a `time.sleep` (en milisegundos)

Observaciones

La sintaxis asume que un `widget` aceptado por el método `.after` ha sido creado previamente (es decir, `widget=tk.Label(parent)`)

Examples

.después()

`.after(delay, callback=None)` es un método definido para todos los widgets tkinter. Este método simplemente llama a la función de `callback` después de la `delay` dada en ms. Si no se proporciona ninguna función, actúa de forma similar a `time.sleep` (pero en milisegundos en lugar de segundos)

Aquí hay un ejemplo de cómo crear un temporizador simple usando `after` :

```
# import tkinter
try:
    import tkinter as tk
except ImportError:
    import Tkinter as tk

class Timer:
    def __init__(self, parent):
        # variable storing time
        self.seconds = 0
        # label displaying time
        self.label = tk.Label(parent, text="0 s", font="Arial 30", width=10)
        self.label.pack()
        # start the timer
        self.label.after(1000, self.refresh_label)

    def refresh_label(self):
```

```
        """ refresh the content of the label every second """
        # increment the time
        self.seconds += 1
        # display the new time
        self.label.configure(text="%i s" % self.seconds)
        # request tkinter to call self.refresh after 1s (the delay is given in ms)
        self.label.after(1000, self.refresh_label)

if __name__ == "__main__":
    root = tk.Tk()
    timer = Timer(root)
    root.mainloop()
```

Lea Retrasando una función en línea: https://riptutorial.com/es/tkinter/topic/6724/retrasando-una-funcion

Capítulo 7: Tkinter Geometry Managers

Introducción

Hay tres administradores de geometría para colocar los widgets: `pack()` , `grid()` y `place()` .

Examples

paquete()

El administrador de geometría de `pack()` organiza los widgets en bloques antes de colocarlos en el widget principal. Utiliza las opciones de `fill` , `expand` y `side` .

Sintaxis

```
widget.pack(option)
```

Llenar
Determina si el widget mantiene el espacio mínimo necesario o ocupa cualquier espacio adicional que se le asigne. Atributos: NINGUNO (predeterminado), X (rellenar horizontalmente), Y (rellenar verticalmente), o AMBOS (rellenar tanto horizontal como verticalmente).

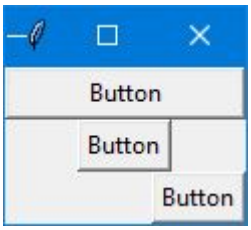
Expandir
Cuando se establece en SÍ, el widget se expande para llenar cualquier espacio no utilizado en el elemento primario del widget. Atributos: SI, NO.

Lado
Determina de qué lado del elemento primario del widget se empaqueta. Atributos: TOP (predeterminado), BOTTOM, LEFT o RIGHT.

Ejemplo

```
from tkinter import *
root = Tk()
btn_fill = Button(root, text="Button")
btn_fill.pack(fill=X)

btn_expand = Button(root, text="Button")
btn_expand.pack(expand=YES)

btn_side = Button(root, text="Button")
btn_side.pack(side=RIGHT)

root.mainloop()
```

Resultado



cuadrícula()

El administrador de geometría `grid()` organiza los widgets en una estructura similar a una tabla en el widget principal. El widget maestro se divide en filas y columnas, y cada parte de la tabla puede contener un widget. Utiliza `column` , `columnspan` , `ipadx` , `ipady` , `padx` , `pady` , `row` , `rowspan` y `sticky` .

Sintaxis

```
widget.grid(options)
```

Columna
La columna para colocar el widget. La columna predeterminada es 0, que es la columna más a la izquierda.

Columnspan
Cuantas widgets de columnas ocupa. El valor predeterminado es 1.

Ipadx
Cuántos píxeles para rellenar el widget horizontalmente dentro de los bordes del widget.

Ipady
Cuántos píxeles para rellenar el widget verticalmente dentro de los bordes del widget.

Padx
Cuántos píxeles para rellenar el widget horizontalmente fuera de los bordes del widget.

Pady
Cuántos píxeles para rellenar el widget verticalmente fuera de los bordes del widget.

Fila
La fila para colocar el widget. La fila predeterminada es 0, que es la columna superior.

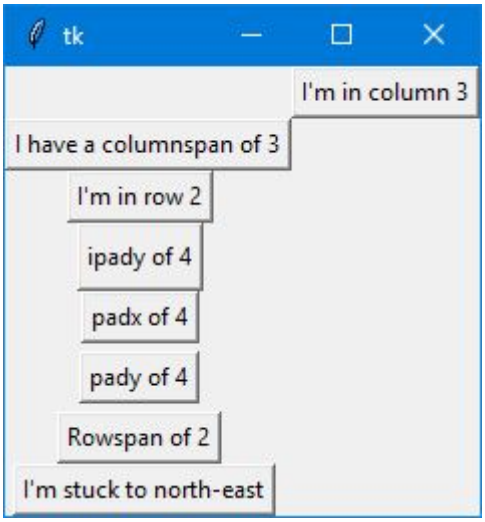
Envergadura
Cuántas filas ocupa el widget. El valor predeterminado es 1.

Pegajoso
Cuando el widget es más pequeño que la celda, `sticky` se usa para indicar a qué lados y esquinas de la celda se adhiere el widget. La dirección se define por las direcciones de la brújula: N, E, S, W, NE, NW, SE y SW y cero. Estas podrían ser una concatenación de cadenas, por ejemplo, NESW hace que el widget ocupe toda el área de la celda.

Ejemplo

```
from tkinter import *
root = Tk()
btn_column = Button(root, text="I'm in column 3")
btn_column.grid(column=3)

btn_columnspan = Button(root, text="I have a columnspan of 3")
btn_columnspan.grid(columnspan=3)

btn_ipadx = Button(root, text="ipadx of 4")
btn_ipadx.grid(ipadx=4)

btn_ipady = Button(root, text="ipady of 4")
btn_ipady.grid(ipady=4)

btn_padx = Button(root, text="padx of 4")
btn_padx.grid(padx=4)

btn_pady = Button(root, text="pady of 4")
btn_pady.grid(pady=4)

btn_row = Button(root, text="I'm in row 2")
btn_row.grid(row=2)

btn_rowspan = Button(root, text="Rowspan of 2")
btn_rowspan.grid(rowspan=2)

btn_sticky = Button(root, text="I'm stuck to north-east")
btn_sticky.grid(sticky=NE)

root.mainloop()
```

Resultado



lugar()

El administrador de `place()` organiza los widgets colocándolos en una posición específica en el widget principal. Este gestor de la geometría utiliza las opciones `anchor` , `bordermode` , `height` , `width` , `relheight` , `relwidth` , `relx` , `rely` , `x` e `y` .

Ancla
Indica donde está anclado el widget. Las opciones son direcciones de la brújula: N, E, S, W, NE,

NW, SE o SW, que se relacionan con los lados y las esquinas del widget principal. El valor predeterminado es NW (la esquina superior izquierda del widget)

Bordermode
Bordermode tiene dos opciones: `INSIDE` , que indica que otras opciones se refieren al interior del padre, (Ignorar los bordes del padre) y `OUTSIDE` , que es lo contrario.

Altura
Especifique la altura de un widget en píxeles.

Anchura
Especifique el ancho de un widget en píxeles.

Relheight
La altura como flotante entre 0.0 y 1.0, como una fracción de la altura del widget principal.

Relwidth
Ancho como flotante entre 0.0 y 1.0, como una fracción del ancho del widget principal.

Relx
Desplazamiento horizontal como flotante entre 0.0 y 1.0, como una fracción del ancho del widget principal.

Confiar
Desplazamiento vertical como flotante entre 0.0 y 1.0, como una fracción de la altura del widget principal.

X
Desplazamiento horizontal en píxeles.

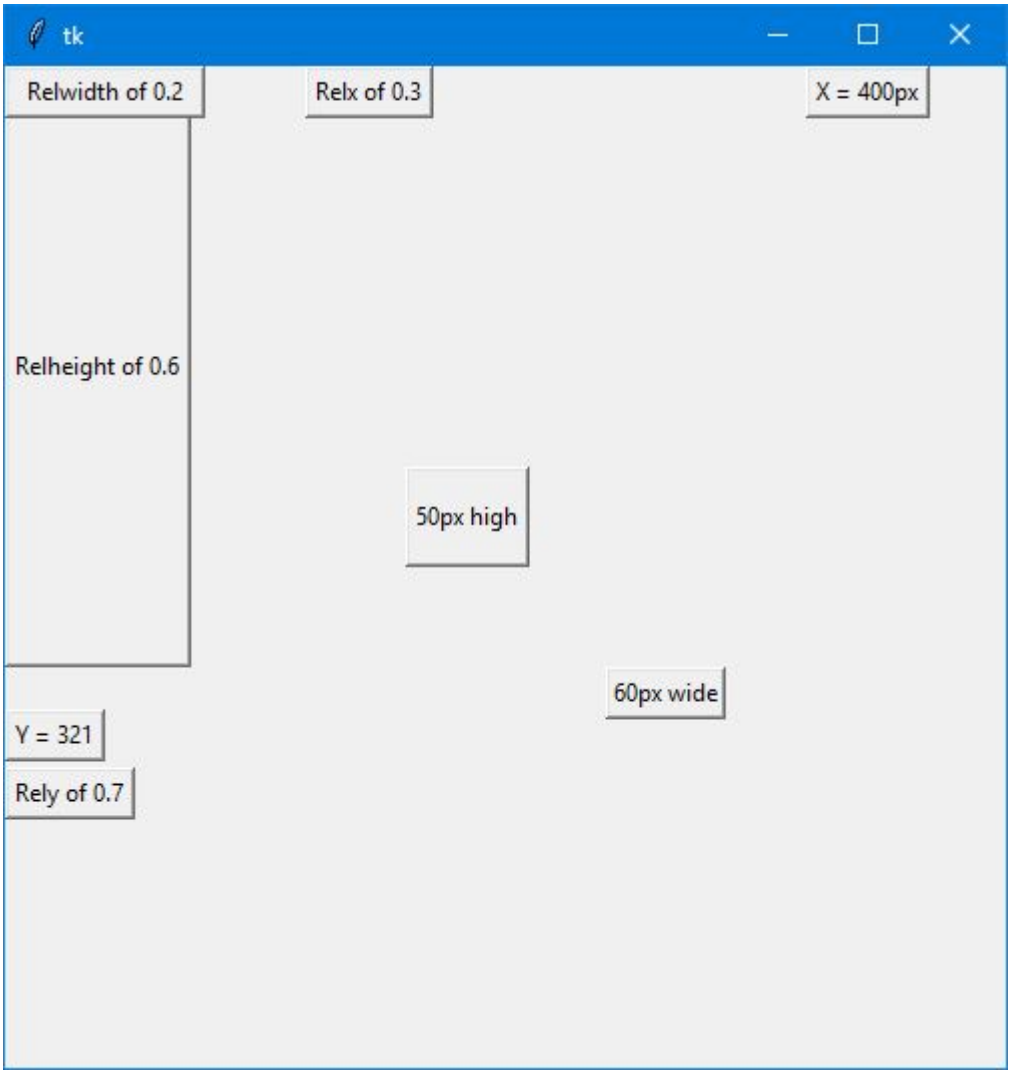
Y
Desplazamiento vertical en píxeles.

Ejemplo

```
from tkinter import *
root = Tk()
root.geometry("500x500")

btn_height = Button(root, text="50px high")
btn_height.place(height=50, x=200, y=200)

btn_width = Button(root, text="60px wide")
btn_width.place(width=60, x=300, y=300)

btn_relheight = Button(root, text="Relheight of 0.6")
btn_relheight.place(relheight=0.6)

btn_relwidth= Button(root, text="Relwidth of 0.2")
btn_relwidth.place(relwidth=0.2)

btn_relx=Button(root, text="Relx of 0.3")
btn_relx.place(relx=0.3)
```

```
btn_rely=Button(root, text="Rely of 0.7")
btn_rely.place(rely=0.7)

btn_x=Button(root, text="X = 400px")
btn_x.place(x=400)

btn_y=Button(root, text="Y = 321")
btn_y.place(y=321)

root.mainloop()
```

Resultado



Lea Tkinter Geometry Managers en línea: https://riptutorial.com/es/tkinter/topic/9620/tkinter-geometry-managers

Capítulo 8: Varias ventanas (widgets TopLevel)

Examples

Diferencia entre Tk y Toplevel

`Tk` es la raíz absoluta de la aplicación, es el primer widget que debe crearse y la GUI se cerrará cuando se destruya.

`Toplevel` es una ventana en la aplicación, cerrar la ventana destruirá todos los widgets secundarios colocados en esa ventana {1} pero no cerrará el programa.

```
try:
    import tkinter as tk #python3
except ImportError:
    import Tkinter as tk #python2

#root application, can only have one of these.
root = tk.Tk()

#put a label in the root to identify the window.
label1 = tk.Label(root, text="""this is root
closing this window will shut down app""")
label1.pack()

#you can make as many Toplevels as you like
extra_window = tk.Toplevel(root)
label2 = tk.Label(extra_window, text="""this is extra_window
closing this will not affect root""")
label2.pack()

root.mainloop()
```

Si su programa de Python solo representa una aplicación única (que casi siempre lo hará), entonces debería tener solo una instancia de `Tk` , pero puede crear tantas ventanas de `Toplevel` como desee.

```
try:
    import tkinter as tk #python3
except ImportError:
    import Tkinter as tk #python2

def generate_new_window():
    window = tk.Toplevel()
    label = tk.Label(window, text="a generic Toplevel window")
    label.pack()

root = tk.Tk()

spawn_window_button = tk.Button(root,
                                text="make a new window!",
```

```
                                command=generate_new_window)
spawn_window_button.pack()

root.mainloop()
```

{1}: si un Toplevel (`A = Toplevel(root)`) es el padre de otro Toplevel (`B = Toplevel(A)`), al cerrar la ventana A también se cerrará la ventana B.

ordenando la pila de ventanas (el método .lift)

El caso más básico para levantar una ventana en particular sobre las otras, simplemente llame al método `.lift()` en esa ventana (ya sea `Toplevel` o `Tk`)

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

root = tk.Tk()

for i in range(4):
    #make a window with a label
    window = tk.Toplevel(root)
    label = tk.Label(window,text="window {}".format(i))
    label.pack()
    #add a button to root to lift that window
    button = tk.Button(root, text = "lift window {}".format(i), command=window.lift)
    button.grid(row=i)

root.mainloop()
```

Sin embargo, si esa ventana se destruye al intentar levantarla, se generará un error como este:

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "/.../tkinter/__init__.py", line 1549, in __call__
    return self.func(*args)
  File "/.../tkinter/__init__.py", line 785, in tkraise
    self.tk.call('raise', self._w, aboveThis)
_tkinter.TclError: bad window path name ".4385637096"
```

A menudo, cuando intentamos colocar una ventana en particular frente al usuario, pero se cerró, una buena alternativa es recrear esa ventana:

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

dialog_window = None

def create_dialog():
    """creates the dialog window
  ** do not call if dialog_window is already open, this will
     create a duplicate without handling the other
if you are unsure if it already exists or not use show_dialog()"""
    global dialog_window
    dialog_window = tk.Toplevel(root)
    label1 = tk.Label(dialog_window,text="this is the dialog window")
    label1.pack()
```

```
    #put other widgets
    dialog_window.lift() #ensure it appears above all others, probably will do this anyway

def show_dialog():
    """lifts the dialog_window if it exists or creates a new one otherwise"""
    #this can be refactored to only have one call to create_dialog()
    #but sometimes extra code will be wanted the first time it is created
    if dialog_window is None:
        create_dialog()
        return
    try:
        dialog_window.lift()
    except tk.TclError:
        #window was closed, create a new one.
        create_dialog()


root = tk.Tk()

dialog_button = tk.Button(root,
                          text="show dialog_window",
                          command=show_dialog)
dialog_button.pack()
root.mainloop()
```

De esta manera, la función `show_dialog` mostrará la ventana de diálogo si existe o no, también tenga en cuenta que puede llamar a `.winfo_exists()` para verificar si existe antes de intentar levantar la ventana en lugar de envolverla en un `try:except` .

También existe el método `.lower()` que funciona de la misma manera que el método `.lift()` , excepto que se baja la ventana en la pila:

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

root = tk.Tk()
root.title("ROOT")
extra = tk.Toplevel()
label = tk.Label(extra, text="extra window")
label.pack()

lower_button = tk.Button(root,
                         text="lower this window",
                         command=root.lower)
lower_button.pack()

root.mainloop()
```

Notará que baja incluso por debajo de otras aplicaciones, para bajar solo por debajo de una determinada ventana, puede pasarlo al método `.lower()` , de manera similar, esto también se puede hacer con el método `.lift()` para elevar una ventana por encima de otra. uno.

Lea Varias ventanas (widgets TopLevel) en línea:
https://riptutorial.com/es/tkinter/topic/6439/varias-ventanas--widgets-toplevel-

Capítulo 9: Widgets de desplazamiento

Introducción

Las barras de desplazamiento se pueden agregar a los widgets Listbox, Canvas y Text. Además, los widgets de entrada se pueden desplazar horizontalmente. Para poder desplazar otro tipo de widgets, debe colocarlos dentro de un lienzo o un widget de texto.

Sintaxis

- scrollbar = tk.Scrollbar (parent, ** kwargs)

Parámetros

Parámetro	Descripción
padre	Los widgets tkinter existen en una jerarquía. A excepción de la ventana raíz, todos los widgets tienen un padre. Algunos tutoriales en línea llaman a esto "maestro". Cuando el widget se agregue a la pantalla con paquete, lugar o cuadrícula, aparecerá dentro de este widget principal
orientar	Orientación de la barra de desplazamiento, ya sea `"vertical"` (valor predeterminado) o `"horizontal"`

Observaciones

Estos ejemplos asumen que tkinter se ha importado con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Examples

Conexión de una barra de desplazamiento vertical a un widget de texto

La conexión entre el widget y la barra de desplazamiento va en ambos sentidos. La barra de desplazamiento debe expandirse verticalmente para que tenga la misma altura que el widget.

```
text = tk.Text(parent)
text.pack(side="left")

scroll_y = tk.Scrollbar(parent, orient="vertical", command=text.yview)
scroll_y.pack(side="left", expand=True, fill="y")

text.configure(yscrollcommand=scroll_y.set)
```

Desplazando un widget de Canvas horizontal y verticalmente

El principio es esencialmente el mismo que para el widget de texto, pero se utiliza un diseño de `Grid` para colocar las barras de desplazamiento alrededor del widget.

```
canvas = tk.Canvas(parent, width=150, height=150)
canvas.create_oval(10, 10, 20, 20, fill="red")
canvas.create_oval(200, 200, 220, 220, fill="blue")
canvas.grid(row=0, column=0)

scroll_x = tk.Scrollbar(parent, orient="horizontal", command=canvas.xview)
scroll_x.grid(row=1, column=0, sticky="ew")

scroll_y = tk.Scrollbar(parent, orient="vertical", command=canvas.yview)
scroll_y.grid(row=0, column=1, sticky="ns")

canvas.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
```

A diferencia del widget de texto, la región desplazable del lienzo no se actualiza automáticamente cuando se modifica su contenido, por lo que debemos definirla y actualizarla manualmente utilizando el argumento `scrollregion` :

```
canvas.configure(scrollregion=canvas.bbox("all"))
```

`canvas.bbox("all")` devuelve las coordenadas del rectángulo que se ajustan a todo el contenido del lienzo.

Desplazando un grupo de widgets

Cuando una ventana contiene muchos widgets, es posible que no todos estén visibles. Sin embargo, ni una ventana (Tk o instancia Toplevel) ni un marco son desplazables. Una solución para hacer que el contenido de la ventana se pueda desplazar es colocar todos los widgets en un marco y luego incrustar este marco en un lienzo utilizando el método `create_window` .

```
canvas = tk.Canvas(parent)
scroll_y = tk.Scrollbar(parent, orient="vertical", command=canvas.yview)

frame = tk.Frame(canvas)
# group of widgets
for i in range(20):
    tk.Label(frame, text='label %i' % i).pack()
# put the frame in the canvas
canvas.create_window(0, 0, anchor='nw', window=frame)
# make sure everything is displayed before configuring the scrollregion
canvas.update_idletasks()

canvas.configure(scrollregion=canvas.bbox('all'),
                 yscrollcommand=scroll_y.set)

canvas.pack(fill='both', expand=True, side='left')
scroll_y.pack(fill='y', side='right')
```

Lea Widgets de desplazamiento en línea: https://riptutorial.com/es/tkinter/topic/8931/widgets-de-

desplazamiento

Capítulo 10: Widgets ttk

Introducción

Ejemplos de los diferentes widgets ttk. Ttk tiene un total de 17 widgets, once de los cuales ya existían en tkinter (tk).

El uso del módulo ttk le da a su aplicación un aspecto más moderno y mejorado.

Sintaxis

- tree = ttk.Treeview (master, ** kwargs)

Parámetros

Parámetro	Descripción
dominar	Los widgets tkinter existen en una jerarquía. Excepto por la ventana raíz, todos los widgets tienen un padre (también llamado "maestro"). Cuando el widget se agregue a la pantalla con paquete, lugar o cuadrícula, aparecerá dentro de este widget principal

Observaciones

Estos ejemplos asumen que tkinter se ha importado con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

También se supone que ttk se ha importado con `from tkinter import ttk` (python 3) o `import ttk` (python 2).

Examples

Treeview: ejemplo básico

Este widget se utiliza para mostrar elementos con jerarquía. Por ejemplo, el explorador de Windows se puede reproducir de esta manera. Algunas bonitas tablas también se pueden hacer usando el widget de vista de `treeview` .

Crear el widget

```
tree=ttk.Treeview(master)
```

Definición de las columnas.

Puede definir cuántas columnas, su ancho y su ancho mínimo cuando el usuario intenta estirarlo. Al definir `stretch=tk.NO` , el usuario no puede modificar el ancho de la columna.

```
tree["columns"]=("one","two","three")
tree.column("#0", width=270, minwidth=270, stretch=tk.NO)
tree.column("one", width=150, minwidth=150, stretch=tk.NO)
tree.column("two", width=400, minwidth=200)
tree.column("three", width=80, minwidth=50, stretch=tk.NO)
```

Definición de los encabezados.

```
tree.heading("#0",text="Name",anchor=tk.W)
tree.heading("one", text="Date modified",anchor=tk.W)
tree.heading("two", text="Type",anchor=tk.W)
tree.heading("three", text="Size",anchor=tk.W)
```

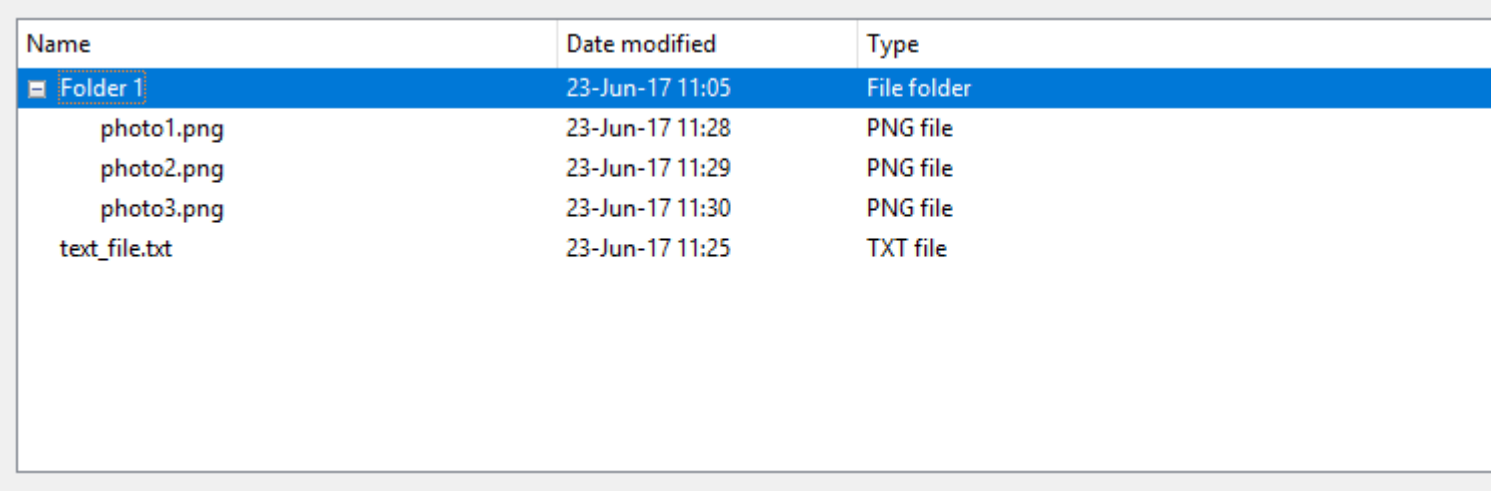
Insertar algunas filas

```
# Level 1
folder1=tree.insert("", 1, "", text="Folder 1", values=("23-Jun-17 11:05","File folder",""))
tree.insert("", 2, "", text="text_file.txt", values=("23-Jun-17 11:25","TXT file","1 KB"))
# Level 2
tree.insert(folder1, "end", "", text="photo1.png", values=("23-Jun-17 11:28","PNG file","2.6
KB"))
tree.insert(folder1, "end", "", text="photo2.png", values=("23-Jun-17 11:29","PNG file","3.2
KB"))
tree.insert(folder1, "end", "", text="photo3.png", values=("23-Jun-17 11:30","PNG file","3.1
KB"))
```

Embalaje

```
tree.pack(side=tk.TOP,fill=tk.X)
```

En Windows, la siguiente captura de pantalla se puede obtener de este ejemplo.



Name	Date modified	Type
Folder 1	23-Jun-17 11:05	File folder
photo1.png	23-Jun-17 11:28	PNG file
photo2.png	23-Jun-17 11:29	PNG file
photo3.png	23-Jun-17 11:30	PNG file
text_file.txt	23-Jun-17 11:25	TXT file

Barra de progreso

El widget `ttk.progress` es útil cuando se trata de cálculos largos para que el usuario sepa que el programa se está ejecutando. A continuación, se muestra un ejemplo de actualización de una barra de progreso cada 0,5 segundos:

Función de actualización de la barra de progreso.

```
def progress(currentValue):
    progressbar["value"]=currentValue
```

Establecer el valor máximo

```
maxValue=100
```

Crea la barra de progreso

```
progressbar=ttk.Progressbar(master,orient="horizontal",length=300,mode="determinate")
progressbar.pack(side=tk.TOP)
```

El modo "determinado" se usa cuando la barra de progreso está bajo el control del programa.

Valores iniciales y máximos.

```
currentValue=0
progressbar["value"]=currentValue
progressbar["maximum"]=maxValue
```

Emular el progreso cada 0.5 s.

```
divisions=10
for i in range(divisions):
    currentValue=currentValue+10
    progressbar.after(500, progress(currentValue))
    progressbar.update() # Force an update of the GUI
```

Lea Widgets ttk en línea: https://riptutorial.com/es/tkinter/topic/10622/widgets-ttk

Creditos

S. No	Capítulos	Contributors
1	Empezando con tkinter	Billal BEGUERADJ, Bryan Oakley, Community, J.J. Hakala, j_4321, JGreenwell, Mike - SMT, Neil A., Nico Brubaker, Razik, ryneke, Tadhg McDonald-Jensen, tao, Yamboy1
2	Agregar imágenes a la etiqueta / botón	Angrywasabi
3	El widget de entrada Tkinter	Angrywasabi, Bryan Oakley, double_j, j_4321
4	El widget Tkinter Radiobutton	j_4321, nbro, Parviz Karimli
5	Personaliza los estilos ttk.	David Duran
6	Retrasando una función	David Duran, Neil A., Tadhg McDonald-Jensen
7	Tkinter Geometry Managers	Henry
8	Varias ventanas (widgets TopLevel)	Tadhg McDonald-Jensen
9	Widgets de desplazamiento	j_4321
10	Widgets ttk	David Duran