EBook Gratuito

APPRENDIMENTO tkinter

Free unaffiliated eBook created from
Stack Overflow contributors.

#tkinter

Sommario

Di

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

Capitolo 1: Iniziare con tkinter

Osservazioni

Tkinter (" **Tk Inter** face") è il pacchetto standard multipiattaforma di python per la creazione di interfacce utente grafiche (GUI). Fornisce l'accesso a un interprete Tcl sottostante con il toolkit Tk, che a sua volta è una libreria di interfaccia grafica multilingue multipiattaforma.

Tkinter non è l'unica libreria GUI per Python, ma è quella che viene fornita come standard. Ulteriori librerie GUI che possono essere usate con python includono wxPython , PyQt e kivy .

La più grande forza di Tkinter è la sua ubiquità e semplicità. Funziona fuori dalla scatola sulla maggior parte delle piattaforme (linux, OSX, Windows) e viene fornito completo di una vasta gamma di widget necessari per le attività più comuni (pulsanti, etichette, disegno su tela, testo multilinea, ecc.).

Come strumento di apprendimento, tkinter ha alcune caratteristiche che sono uniche tra i toolkit della GUI, come i font nominati, i tag di binding e la tracciabilità variabile.

Differenze tra Python 2 e 3

Tkinter è in gran parte invariato tra python 2 e python 3, con la differenza che il pacchetto e i moduli tkinter sono stati rinominati.

Importazione in python 2.x

In python 2.x, il pacchetto tkinter si chiama `Tkinter` e i pacchetti correlati hanno i loro nomi. Ad esempio, quanto segue mostra un insieme tipico di istruzioni di importazione per python 2.x:

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

Importazione in python 3.x

Sebbene le funzionalità non cambino molto tra Python 2 e 3, i nomi di tutti i moduli di tkinter sono cambiati. Di seguito è riportato un set tipico di istruzioni di importazione per python 3.x:

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

Ulteriori letture

- Domande di Tkinter su Stackoverflow
- Documentazione ufficiale di tkinter di Python 3
- Documentazione ufficiale di tkinter di Python 2
- Tkdocs.com - documentazione tk multipiattaforma
- Introduzione di Effbot a tkinter
- Guida di riferimento di Tkinter, New Mexico Tech

Versioni

Tcl

Versione	Data di rilascio
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

Pitone

Versione	Data di rilascio
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
3.1	2009-06-26
3.0	2008-12-03
2.7	2010-07-03

Versione	Data di rilascio
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

Installazione o configurazione

Tkinter viene preinstallato con i binari di installazione di Python per Mac OS X e la piattaforma Windows. Quindi, se installi Python dai binari ufficiali per Mac OS X o la piattaforma Windows, sei a posto con Tkinter.

Per le versioni Debian di Linux devi installarlo manualmente usando i seguenti comandi.

Per Python 3

    sudo apt-get install python3-tk

Per Python 2.7

    sudo apt-get installa python-tk

Le distribuzioni Linux con yum installer possono installare il modulo tkinter usando il comando:

    yum install tkinter

Verifica l'installazione

Per verificare se hai installato correttamente Tkinter, apri la tua console Python e digita il seguente comando:

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

o

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

Tkinter è stato installato correttamente, se il comando precedente viene eseguito senza errori.

Per verificare la versione di Tkinter, digitare i seguenti comandi nel REPL di Python:

Per python 3.X

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

Per python 2.X

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

Nota: l'importazione di `Tkinter as tk` non è richiesta ma è una buona pratica in quanto aiuta a mantenere le cose coerenti tra la versione.

Ciao mondo! (minimo)

Mettiamo alla prova la nostra conoscenza di base di tkinter creando il classico "Hello, World!" programma.

Per prima cosa, dobbiamo importare tkinter, questo varierà in base alla versione (vedere la sezione commenti su "Differenze tra Python 2 e 3")

In Python 3 il modulo `tkinter` ha una t minuscola:

```
import tkinter as tk
```

In Python 2 il modulo `Tkinter` ha una T maiuscola:

```
import Tkinter as tk
```

L'utilizzo di `as tk` non è strettamente necessario, ma lo useremo, quindi il resto di questo esempio funzionerà allo stesso modo per entrambe le versioni.

ora che abbiamo il modulo tkinter importato possiamo creare la root della nostra applicazione usando la classe `Tk` :

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

Questo fungerà da finestra per la nostra applicazione. (nota che le finestre *aggiuntive* dovrebbero essere istanze di `Toplevel`)

Ora che abbiamo una finestra, aggiungiamo del testo con `Label`

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

Una volta che l'applicazione è pronta, possiamo avviarla (inserire il *ciclo degli* eventi *principale*) con il metodo `mainloop`

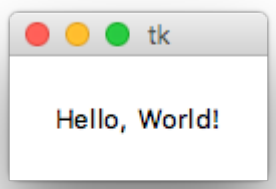
```
root.mainloop()
```

Questo si aprirà ed eseguirà l'applicazione fino a quando non viene interrotta dalla chiusura della finestra o chiamando le funzioni di uscita dai callback (discussi in seguito) come `root.destroy()` .

Mettere tutto insieme:

```
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()
```

E qualcosa del genere dovrebbe apparire:



Ciao mondo! (modulare, orientato agli oggetti)

```
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: è possibile ereditare da qualsiasi widget tkinter, inclusa la finestra radice. Ereditare da `tkinter.Frame` è almeno il più flessibile in quanto supporta più interfacce di documenti (MDI), singole interfacce di documenti (SDI), applicazioni a singola pagina e applicazioni a più pagine.

Leggi Iniziare con tkinter online: https://riptutorial.com/it/tkinter/topic/987/iniziare-con-tkinter

Capitolo 2: Aggiunta di immagini a etichetta / pulsante

introduzione

Questo mostra l'uso corretto delle immagini e come visualizzare correttamente le immagini.

Examples

Formati di file supportati da Tkinter

File di supporto .ppm di Tkinter da PIL (Python Imaging Library), .JPG, .PNG e .GIF.

Per importare e immagine devi prima creare un riferimento in questo modo:

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

Ora, possiamo aggiungere questa immagine a Button ed Etichette in questo modo usando il callback "img":

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

Utilizzo dei formati .GIF.

Per visualizzare una gif, devi mostrarla fotogramma per fotogramma come un'animazione.

Una gif animata consiste in un numero di frame in un singolo file. Tk carica il primo fotogramma ma è possibile specificare diversi fotogrammi passando un parametro indice durante la creazione dell'immagine. Per esempio:

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

Se caricate tutti i fotogrammi in PhotoImages separati e quindi utilizzate gli eventi del timer per cambiare la cornice visualizzata (label.configure (image = nextframe)). Il ritardo sul timer consente di controllare la velocità di animazione. Non è fornito nulla per darti il numero di fotogrammi nell'immagine oltre a non riuscire a creare una cornice una volta superato il numero di fotogrammi.

Leggi Aggiunta di immagini a etichetta / pulsante online:
https://riptutorial.com/it/tkinter/topic/9746/aggiunta-di-immagini-a-etichetta---pulsante

Capitolo 3: Finestre multiple (widget TopLevel)

Examples

Differenza tra Tk e Toplevel

`Tk` è la radice assoluta dell'applicazione, è il primo widget che deve essere istanziato e la GUI si spegne quando viene distrutta.

`Toplevel` è una finestra dell'applicazione, chiudendo la finestra si distruggeranno tutti i widget figli posizionati su quella finestra {1} ma non si spegnerà il programma.

```
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()
```

Se il tuo programma python rappresenta solo una singola applicazione (cosa che quasi sempre `Toplevel`) dovresti avere una sola istanza di `Tk` , ma puoi creare `Toplevel` finestre `Toplevel` che desideri.

```
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}: se un Toplevel (`A = Toplevel(root)`) è il genitore di un altro Toplevel (`B = Toplevel(A)`), la finestra di chiusura A chiuderà anche la finestra B.

organizzare la pila di finestre (il metodo .lift)

Il caso più semplice per sollevare una finestra particolare sopra gli altri, basta chiamare il metodo `.lift()` su quella finestra (`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()
```

Tuttavia, se quella finestra viene distrutta cercando di sollevarla, si genera un errore come questo:

```
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"
```

Spesso, quando proviamo a mettere una finestra particolare davanti all'utente, ma è stata chiusa, una buona alternativa è ricreare quella finestra:

```
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()
```

In questo modo la funzione `show_dialog` mostrerà alla finestra di dialogo se esiste o meno, inoltre si noti che è possibile chiamare `.winfo_exists()` per verificare se esiste prima di provare a sollevare la finestra invece di avvolgerla in una `try:except` .

Esiste anche il metodo `.lower()` che funziona allo stesso modo del metodo `.lift()` , ad eccezione dell'abbassare la finestra nello stack:

```
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()
```

Noterai che si abbassa anche al di sotto di altre applicazioni, per abbassare solo al di sotto di una certa finestra puoi passarlo al metodo `.lower()` , allo stesso modo questo può essere fatto anche con il metodo `.lift()` per sollevare solo una finestra sopra un'altra uno.

Leggi Finestre multiple (widget TopLevel) online:
https://riptutorial.com/it/tkinter/topic/6439/finestre-multiple--widget-toplevel-

Capitolo 4: Il Tkinter Entry Widget

Sintassi

- entry = tk.Entry (*parent* , ** *kwargs*)
- entry.get ()
- entry.insert (index, "valore")
- entry.delete (start_index, end_index)
- entry.bind (evento, callback)

Parametri

Parametro	Descrizione
genitore	i widget tkinter esistono in una struttura gerarchica. Ad eccezione della finestra radice, tutti i widget hanno un genitore. Alcuni tutorial online chiamano questo "master". Quando il widget viene aggiunto allo schermo con `pack` , `place` o `grid` , apparirà all'interno di questo widget genitore
larghezza	La larghezza specifica la larghezza *desiderata* del widget in base alla larghezza media di un carattere. Per i caratteri a larghezza variabile, questo è basato sulla larghezza del carattere zero (`0`). Il valore predefinito è 20. Si noti che la larghezza effettiva potrebbe essere maggiore o minore a seconda di come viene aggiunta allo schermo.

Osservazioni

Questi esempi presuppongono che tkinter sia stato importato con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Examples

Creazione di un widget di voce e impostazione di un valore predefinito

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

Ottenere il valore di un widget Entry

Il valore di un widget voce può essere ottenuto con il metodo `get` del widget:

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

```
name = name_entry.get()
```

Facoltativamente, è possibile associare un'istanza di `StringVar` e recuperare il valore dallo `StringVar` anziché dal widget:

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

Aggiunta di convalida a un widget di voce

Per limitare i caratteri che possono essere digitati in un widget di voci, solo i numeri, ad esempio, un comando di convalida può essere aggiunto alla voce. Un comando di convalida è una funzione che restituisce `True` se la modifica è accettata, `False` altrimenti. Questa funzione sarà chiamata ogni volta che il contenuto della voce viene modificato. Vari argomenti possono essere passati a questa funzione, come il tipo di modifica (inserimento, cancellazione), il testo inserito, ...

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

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

L'opzione di `validate` determina il tipo di evento che attiva la convalida, in questo caso è presente un qualsiasi tasto nella voce. Il `'%S'` nell'opzione validatecommand significa che il carattere inserito o eliminato viene passato in argomento alla funzione `only_numbers` . L'elenco completo delle possibilità può essere trovato qui .

Getting int From Entry Widget

Quando si usa il metodo .get (), qualunque cosa sia nel widget di voce verrà convertito in una stringa. Ad esempio, indipendentemente dal tipo di input (può essere un numero o una frase), il risultato risultante sarà una stringa. Se l'utente digita 4 l'output sarà "4" come in una stringa. Per ottenere un int da un Entry Widget, prima chiama il metodo .get ().

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

Ora convertiamo quella stringa in un int come:

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

Allo stesso modo, se vuoi risparmiare tempo puoi semplicemente fare:

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

È possibile utilizzare il metodo precedente se non si desidera convertire str in int.

Leggi Il Tkinter Entry Widget online: https://riptutorial.com/it/tkinter/topic/4868/il-tkinter-entry-widget

Capitolo 5: Il widget Tkinter Radiobutton

Sintassi

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

Parametri

Parametro	Descrizione
genitore	i widget tkinter esistono in una gerarchia. Ad eccezione della finestra radice, tutti i widget hanno un genitore. Alcuni tutorial online chiamano questo "master". Quando il widget viene aggiunto allo schermo con pack, luogo o griglia, apparirà all'interno di questo widget genitore.
comando	funzione chiamata ogni volta che l'utente cambia lo stato del radiobutton
indicatoron	1 o True per i pulsanti di opzione, 0 o False per i pulsanti
testo	Testo da visualizzare accanto al pulsante radio.
valore	Quando viene selezionato il radio button, la variabile di controllo associata viene impostata sul valore.
variabile	Controlla la variabile che il radiobutton condivide con l'altro radiobutton del gruppo.

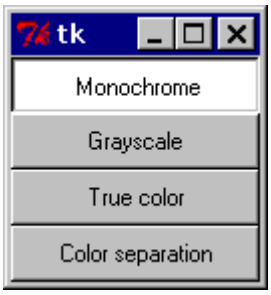
Osservazioni

Questi esempi presuppongono che tkinter sia stato importato con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Riferimento:

Per trasformare l'esempio sopra in una "scatola dei pulsanti" piuttosto che in un set di pulsanti di opzione, imposta l'opzione dell'indicatore su 0. In questo caso, non c'è un indicatore di pulsante radio separato, e il pulsante selezionato viene disegnato come

SUNKEN anziché RAISED:



- effbot

Examples

Ecco un esempio di come attivare i pulsanti di opzione nelle caselle dei pulsanti:

```
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()
```

Crea un gruppo di radiobutton

Tale gruppo è costituito da pulsanti radio che condividono una variabile di controllo in modo che non sia possibile selezionarne più di 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()
```

Leggi Il widget Tkinter Radiobutton online: https://riptutorial.com/it/tkinter/topic/6338/il-widget-tkinter-radiobutton

Capitolo 6: Personalizza gli stili TTK

introduzione

Lo stile dei nuovi widget ttk è uno degli aspetti più potenti di ttk. Oltre al fatto che è un modo di lavorare completamente diverso rispetto al tradizionale pacchetto tk, consente di eseguire un enorme grado di personalizzazione sui tuoi widget.

Examples

Personalizza una vista ad albero

Prendendo Treeview: esempio di base , può essere mostrato come personalizzare una vista ad albero di base.

In questo caso, creiamo uno stile "mystyle.Treeview" con il seguente codice (vedi i commenti per capire cosa fa ogni riga):

```
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
```

Quindi, il widget viene creato dando lo stile di cui sopra:

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

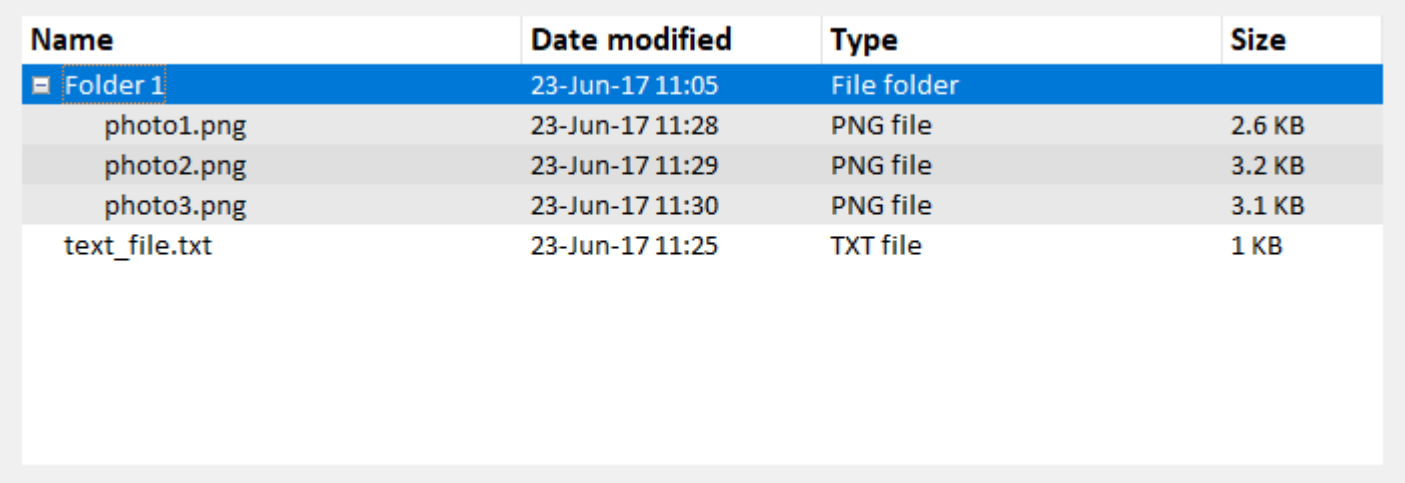
Se desideri avere un formato diverso a seconda delle righe, puoi utilizzare i `tags` :

```
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',))
```

Quindi, ad esempio, un colore di sfondo può essere associato ai tag:

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

Il risultato è una vista ad albero con caratteri modificati sul corpo e sulle intestazioni, senza bordi e colori diversi per le righe:



Nota: per generare l'immagine sopra, è necessario aggiungere / modificare le linee di codice summenzionate nell'esempio Vista di esempio : esempio di base .

Leggi Personalizza gli stili TTK online: https://riptutorial.com/it/tkinter/topic/10624/personalizza-gli-stili-ttk

Capitolo 7: Ritardare una funzione

Sintassi

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

Parametri

Parametro	Descrizione
delay_ms	Tempo (millisecondi), che è in ritardo la chiamata alla funzione `callback`
richiama	Funzione chiamata dopo il dato `delay_ms` . Se questo parametro non viene fornito, `.after` agisce in modo simile a `time.sleep` (in millisecondi)

Osservazioni

La sintassi presuppone un `widget` accettato dal metodo. `.after` è stato precedentemente creato (ad es. `widget=tk.Label(parent)`)

Examples

.dopo()

`.after(delay, callback=None)` è un metodo definito per tutti i widget tkinter. Questo metodo chiama semplicemente la funzione `callback` dopo il dato `delay` in ms. Se non viene fornita alcuna funzione, agisce in modo simile a `time.sleep` (ma in millisecondi anziché secondi)

Ecco un esempio di come creare un semplice timer 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()
```

Leggi Ritardare una funzione online: https://riptutorial.com/it/tkinter/topic/6724/ritardare-una-funzione

Capitolo 8: Tkinter Geometry Managers

introduzione

Esistono tre gestori di geometria per posizionare i widget: `pack()` , `grid()` e `place()` .

Examples

pack ()

Il gestore della geometria `pack()` organizza i widget in blocchi prima di posizionarli nel widget padre. Usa le opzioni `fill` , `expand` e `side` .

Sintassi

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

Riempire
Determina se il widget mantiene lo spazio minimo necessario o occupa uno spazio aggiuntivo ad esso assegnato. Attributi: NONE (predefinito), X (riempimento orizzontale), Y (riempimento verticale) o BOTH (riempimento sia in senso orizzontale che verticale).

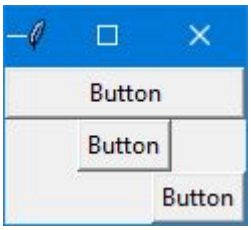
Espandere
Se impostato su SÌ, il widget si espande per riempire qualsiasi spazio non utilizzato nel genitore del widget. Attributi: SÌ, NO.

Lato
Determina il lato del genitore del widget a cui è impacchettato. Attributi: TOP (predefinito), BOTTOM, SINISTRA o DESTRA.

Esempio

```
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()
```

Risultato



griglia()

Il gestore della geometria `grid()` organizza i widget in una struttura simile a una tabella nel widget principale. Il widget principale è suddiviso in righe e colonne e ciascuna parte della tabella può contenere un widget. Usa `column` , `columnspan` , `ipadx` , `ipady` , `padx` , `pady` , `row` , `rowspan` e `sticky` .

Sintassi

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

Colonna
La colonna in cui inserire il widget. La colonna predefinita è 0, che è la colonna più a sinistra.

columnspan
Quante widget di colonne occupa. Il valore predefinito è 1.

Ipadx
Quanti pixel riempiono orizzontalmente il widget all'interno dei bordi del widget.

Ipady
Quanti pixel riempiono verticalmente il widget all'interno dei bordi del widget.

padx
Quanti pixel riempiono il widget orizzontalmente al di fuori dei bordi del widget.

Pady
Quanti pixel riempiono verticalmente il widget all'esterno dei bordi del widget.

Riga
La riga in cui inserire il widget. La riga predefinita è 0, che è la colonna più in alto.

rowspan
Quante righe occupa il widget. Il valore predefinito è 1.

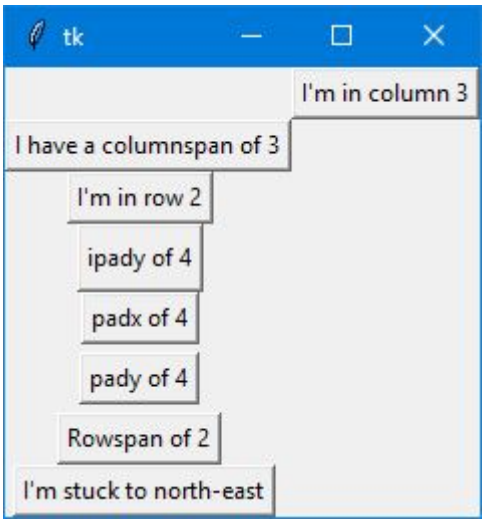
Appiccicoso
Quando il widget è più piccolo della cella, viene utilizzato `sticky` per indicare a quali lati e angoli della cella si attacca il widget. La direzione è definita dalle direzioni della bussola: N, E, S, W, NE, NW, SE e SW e zero. Questi potrebbero essere una concatenazione di stringhe, ad esempio NESW fa in modo che il widget occupi l'intera area della cella.

Esempio

```
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()
```

Risultato



posto()

Il gestore `place()` organizza i widget posizionandoli in una posizione specifica nel widget principale. Questo gestore di geometria utilizza le opzioni `anchor` , `bordermode` , `height` , `width` , `relheight` , `relwidth` , `relx` , `rely` , `x` e `y` .

Ancora
Indica dove è ancorato il widget. Le opzioni sono le direzioni della bussola: N, E, S, W, NE, NW, SE o SW, che si riferiscono ai lati e agli angoli del widget principale. L'impostazione predefinita è

NW (l'angolo in alto a sinistra del widget)

Bordermode
Bordermode ha due opzioni: `INSIDE` , che indica che le altre opzioni si riferiscono ai genitori all'interno, (Ignorando i bordi del genitore) e `OUTSIDE` , che è il contrario.

Altezza
Specificare l'altezza di un widget in pixel.

Larghezza
Specificare la larghezza di un widget in pixel.

Relheight
Altezza come float tra 0.0 e 1.0, come frazione dell'altezza del widget genitore.

Relwidth
Larghezza come float tra 0.0 e 1.0, come frazione della larghezza del widget genitore.

relx
Offset orizzontale come float tra 0.0 e 1.0, come frazione della larghezza del widget genitore.

fare affidamento
Offset verticale come float tra 0.0 e 1.0, come frazione dell'altezza del widget genitore.

X
Offset orizzontale in pixel.

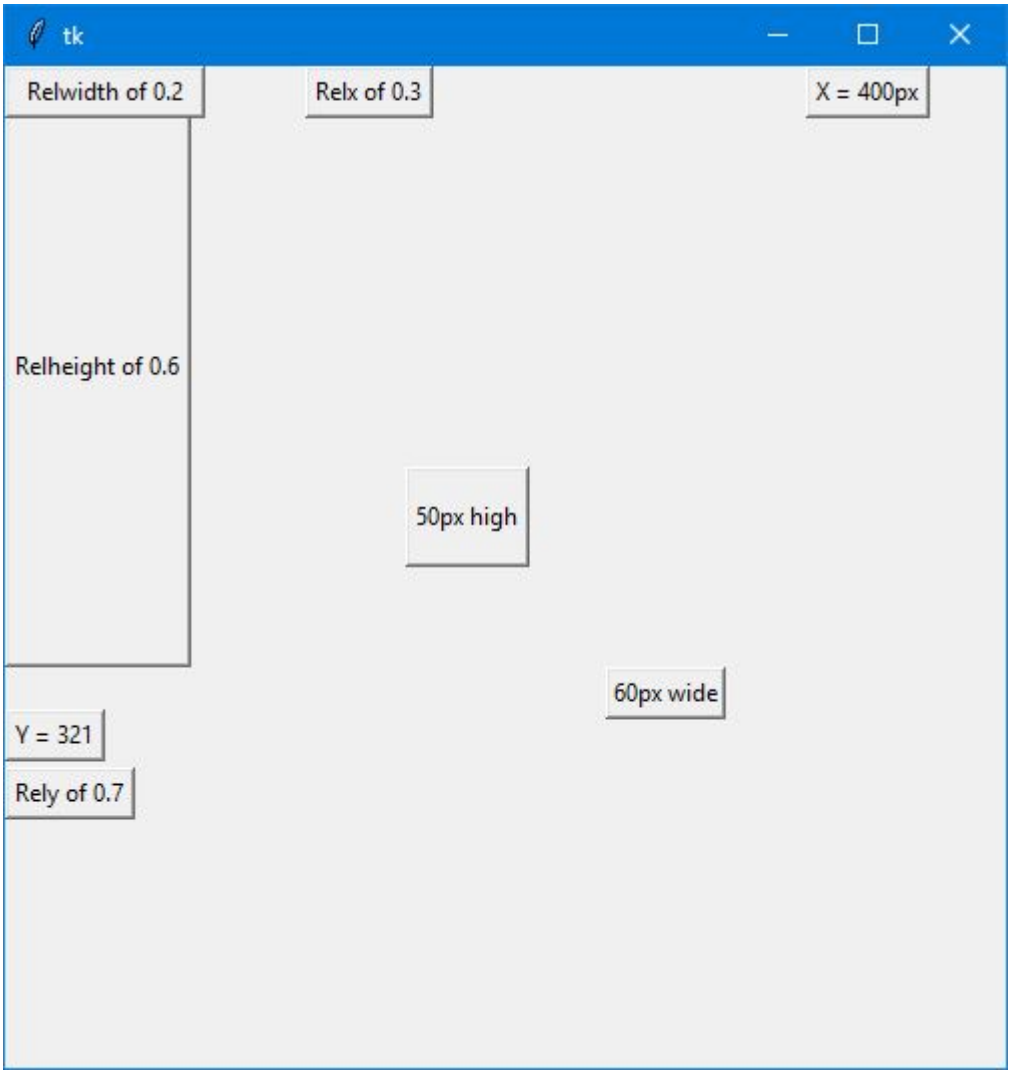
Y
Offset verticale in pixel.

Esempio

```
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()
```

Risultato



Leggi Tkinter Geometry Managers online: https://riptutorial.com/it/tkinter/topic/9620/tkinter-geometry-managers

Capitolo 9: Widget a scorrimento

introduzione

Le barre di scorrimento possono essere aggiunte ai widget Listbox, Canvas e Text. Inoltre, i widget Entry possono essere scrollati orizzontalmente. Per poter scorrere altri tipi di widget, è necessario inserirli in un oggetto Canvas o in un widget Testo.

Sintassi

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

Parametri

Parametro	Descrizione
genitore	i widget tkinter esistono in una gerarchia. Ad eccezione della finestra radice, tutti i widget hanno un genitore. Alcuni tutorial online chiamano questo "master". Quando il widget viene aggiunto allo schermo con pack, luogo o griglia, apparirà all'interno di questo widget genitore
Oriente	Orientamento della barra di scorrimento, `"vertical"` (valore predefinito) o `"horizontal"`

Osservazioni

Questi esempi presuppongono che tkinter sia stato importato con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Examples

Collegamento di una barra di scorrimento verticale a un widget di testo

La connessione tra il widget e la barra di scorrimento va in entrambe le direzioni. La barra di scorrimento deve essere espansa verticalmente in modo che abbia la stessa altezza del 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)
```

Scorrimento di un widget Canvas in senso orizzontale e verticale

Il principio è essenzialmente lo stesso del widget Testo, ma un layout `Grid` viene utilizzato per posizionare le barre di scorrimento attorno al 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 differenza del widget Testo, la regione scrollabile della Tela non viene aggiornata automaticamente quando il suo contenuto viene modificato, quindi dobbiamo definirlo e aggiornarlo manualmente usando l'argomento `scrollregion` :

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

`canvas.bbox("all")` restituisce le coordinate del rettangolo adattando l'intero contenuto del canvas.

Scorrimento di un gruppo di widget

Quando una finestra contiene molti widget, potrebbero non essere tutti visibili. Tuttavia, né una finestra (istanza Tk o Toplevel) né una cornice sono scorrevoli. Una soluzione per rendere scorrevole il contenuto della finestra consiste nel mettere tutti i widget in una cornice e quindi incorporare questa cornice in una tela utilizzando il metodo `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')
```

Leggi Widget a scorrimento online: https://riptutorial.com/it/tkinter/topic/8931/widget-a-scorrimento

Capitolo 10: Widget Ttk

introduzione

Esempi dei diversi widget ttk. Ttk ha un totale di 17 widget, undici dei quali già esistenti in tkinter (tk).

L'uso del modulo ttk conferisce alla tua applicazione un aspetto più moderno e migliorato.

Sintassi

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

Parametri

Parametro	Descrizione
maestro	i widget tkinter esistono in una struttura gerarchica. Ad eccezione della finestra radice, tutti i widget hanno un genitore (chiamato anche "master"). Quando il widget viene aggiunto allo schermo con pack, luogo o griglia, apparirà all'interno di questo widget genitore

Osservazioni

Questi esempi presuppongono che tkinter sia stato importato con `import tkinter as tk` (python 3) o `import Tkinter as tk` (python 2).

Si presume inoltre che ttk sia stato importato `from tkinter import ttk` (python 3) o `import ttk` (python 2).

Examples

Treeview: esempio di base

Questo widget viene utilizzato per visualizzare elementi con gerarchia. Ad esempio, Windows Explorer può essere riprodotto in questo modo. Alcuni bei tavoli possono essere fatti anche usando il widget `treeview` .

Crea il widget

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

Definizione delle colonne

Puoi definire quante colonne, la loro larghezza e larghezza minima quando l'utente prova ad allungarlo. Definendo `stretch=tk.NO` , l'utente non può modificare la larghezza della colonna.

```
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)
```

Definizione delle intestazioni

```
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)
```

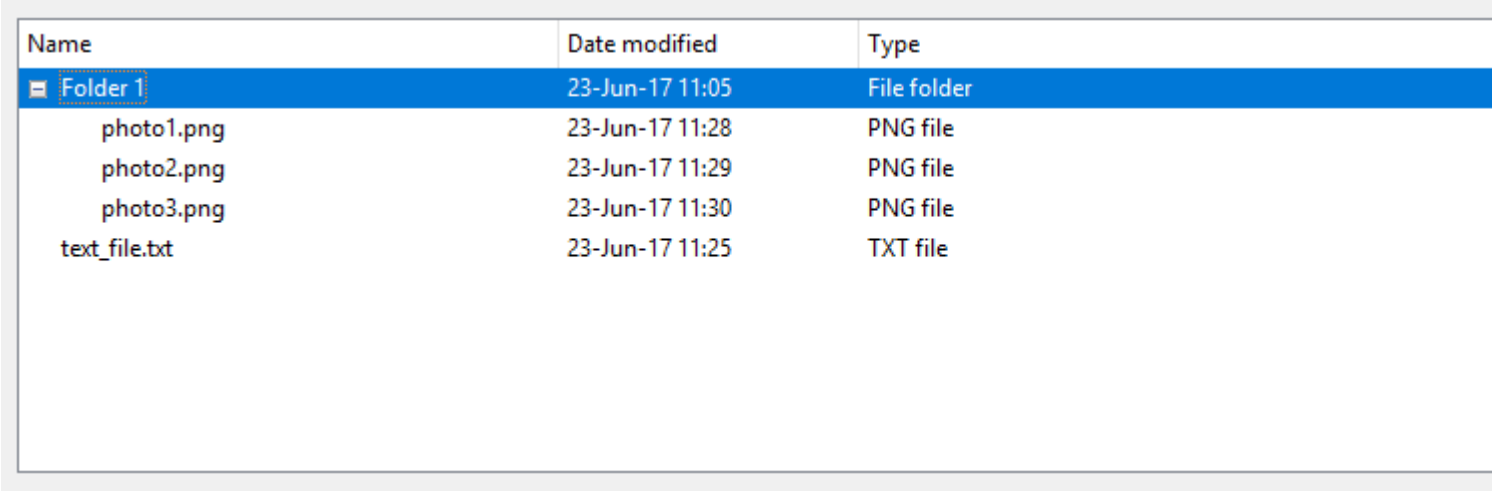
Inserisci alcune righe

```
# 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"))
```

Imballaggio

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

Su Windows, il seguente screenshot può essere ottenuto da questo esempio.



Barra di avanzamento

Il widget `ttk.progress` è utile quando si ha a che fare con calcoli lunghi in modo che l'utente sappia che il programma è in esecuzione. Di seguito, viene fornito un esempio di aggiornamento di una barra di avanzamento ogni 0,5 secondi:

Funzione che aggiorna la barra di avanzamento

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

Imposta il valore massimo

```
maxValue=100
```

Crea la barra di avanzamento

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

La modalità "determinata" viene utilizzata quando la barra di avanzamento è sotto il controllo del programma.

Valori iniziali e massimi

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

```
progressbar["maximum"]=maxValue
```

Emula progresso ogni 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
```

Leggi Widget Ttk online: https://riptutorial.com/it/tkinter/topic/10622/widget-ttk

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare 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	Aggiunta di immagini a etichetta / pulsante	Angrywasabi
3	Finestre multiple (widget TopLevel)	Tadhg McDonald-Jensen
4	Il Tkinter Entry Widget	Angrywasabi, Bryan Oakley, double_j, j_4321
5	Il widget Tkinter Radiobutton	j_4321, nbro, Parviz Karimli
6	Personalizza gli stili TTK	David Duran
7	Ritardare una funzione	David Duran, Neil A., Tadhg McDonald-Jensen
8	Tkinter Geometry Managers	Henry
9	Widget a scorrimento	j_4321
10	Widget Ttk	David Duran