eBook Gratuit

APPRENEZ tkinter

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#tkinter

Table des matières

À propos

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

Chapitre 1: Démarrer avec tkinter

Remarques

Tkinter (" **Tk Inter** face") est le package multi-plateforme standard de python pour la création d'interfaces utilisateur graphiques (GUI). Il donne accès à un interpréteur Tcl sous-jacent avec la boîte à outils Tk, qui est elle-même une bibliothèque d'interface utilisateur graphique multi-plateforme et multilingue.

Tkinter n'est pas la seule bibliothèque graphique pour python, mais c'est celle qui est fournie en standard. Les bibliothèques GUI supplémentaires pouvant être utilisées avec python incluent wxPython , PyQt et kivy .

La plus grande force de Tkinter réside dans son omniprésence et sa simplicité. Il fonctionne directement sur la plupart des plates-formes (Linux, OSX, Windows) et comprend un large éventail de widgets nécessaires pour les tâches les plus courantes (boutons, libellés, dessin, texte multiligne, etc.).

En tant qu'outil d'apprentissage, tkinter possède des fonctionnalités uniques parmi les kits d'outils d'interface graphique, telles que les polices nommées, les balises de liaison et le suivi des variables.

Différences entre python 2 et 3

Tkinter est pratiquement inchangé entre python 2 et python 3, la principale différence étant que le paquet tkinter et les modules ont été renommés.

Importation dans python 2.x

Dans python 2.x, le package tkinter s'appelle `Tkinter` et les packages associés ont leurs propres noms. Par exemple, le tableau suivant montre un jeu d'instructions d'importation typique pour python 2.x:

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

Importation en python 3.x

Bien que la fonctionnalité n'ait pas beaucoup changé entre python 2 et 3, les noms de tous les modules tkinter ont changé. Voici un ensemble typique d'instructions d'importation pour python 3.x:

```
import tkinter as tk
```

```
from tkinter import filedialog
from tkinter import ttk
```

Lectures complémentaires

- Questions Tkinter sur Stackoverflow
- Documentation officielle Python 3 tkinter
- Documentation officielle Python 2 tkinter
- Tkdocs.com - documentation multiplateforme tk
- Effbot introduction à tkinter
- Guide de référence Tkinter, New Mexico Tech

Versions

Tcl

Version	Date de sortie
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

Python

Version	Date de sortie
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

Version	Date de sortie
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

Installation ou configuration

Tkinter est préinstallé avec les fichiers binaires d'installation de Python pour Mac OS X et la plate-forme Windows. Donc, si vous installez Python à partir des binaires officiels pour la plate-forme Mac OS X ou Windows, vous êtes prêt à utiliser Tkinter.

Pour les versions Debian de Linux, vous devez l'installer manuellement en utilisant les commandes suivantes.

Pour Python 3

    sudo apt-get installe python3-tk

Pour Python 2.7

    sudo apt-get installer python-tk

Les distributions Linux avec l'installateur yum peuvent installer le module tkinter à l'aide de la commande:

    yum installer tkinter

Installation en cours de vérification

Pour vérifier si vous avez correctement installé Tkinter, ouvrez votre console Python et tapez la commande suivante:

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

ou

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

Vous avez installé avec succès Tkinter, si la commande ci-dessus s'exécute sans erreur.

Pour vérifier la version de Tkinter, tapez les commandes suivantes dans votre Python REPL:

Pour python 3.X

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

Pour python 2.X

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

Remarque: L'importation de `Tkinter as tk` n'est pas requise, mais c'est une bonne pratique car elle permet de maintenir la cohérence entre les versions.

Bonjour le monde! (minimal)

Testons nos connaissances de base de tkinter en créant le classique "Hello, World!" programme.

D'abord, il faut importer tkinter, cela variera selon la version (voir la section remarques sur "Différences entre Python 2 et 3")

Dans Python 3, le module `tkinter` a un t minuscule:

```
import tkinter as tk
```

Dans Python 2, le module `Tkinter` a un T majuscule:

```
import Tkinter as tk
```

L'utilisation `as tk` n'est pas strictement nécessaire, mais nous allons l'utiliser pour que le reste de cet exemple fonctionne de la même manière pour les deux versions.

Maintenant que le module tkinter est importé, nous pouvons créer la racine de notre application en utilisant la classe `Tk` :

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

Cela servira de fenêtre pour notre application. (notez que *les* fenêtres *supplémentaires* doivent être des instances de `Toplevel` place)

Maintenant que nous avons une fenêtre, ajoutons-y du texte avec une `Label`

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

Une fois que l'application est prête, nous pouvons la démarrer (entrez la *boucle d'* événement *principal*) avec la méthode `mainloop`

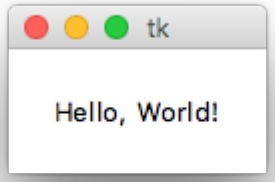
```
root.mainloop()
```

Cela ouvrira et exécutera l'application jusqu'à ce qu'elle soit arrêtée par la fermeture de la fenêtre ou par l'appel de fonctions existantes à partir de rappels (décrits plus loin), tels que `root.destroy()` .

Mettre tous ensemble:

```
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()
```

Et quelque chose comme ça devrait apparaître:



Bonjour le monde! (modulaire, orienté objet)

```
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()
```

Note: Il est possible d'hériter de presque n'importe quel widget tkinter, y compris la fenêtre racine. L'héritage de `tkinter.Frame` est sans doute le plus flexible en ce sens qu'il prend en charge plusieurs interfaces de document (MDI), des interfaces de document unique (SDI), des applications à page unique et des applications à plusieurs pages.

Lire Démarrer avec tkinter en ligne: https://riptutorial.com/fr/tkinter/topic/987/demarrer-avec-tkinter

Chapitre 2: Ajout d'images à une étiquette / bouton

Introduction

Cela montre l'utilisation correcte des images et comment afficher correctement les images.

Examples

Formats de fichiers pris en charge par Tkinter

Tkinter supporte les fichiers .ppm de PIL (Python Imaging Library), .JPG, .PNG et .GIF.

Pour importer et créer une image, vous devez d'abord créer une référence comme ceci:

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

Maintenant, nous pouvons ajouter cette image à Button et Labels comme cela en utilisant le callback "img":

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

Utilisation des formats .GIF

Pour afficher un gif, vous devez l'afficher image par image, comme une animation.

Un gif animé se compose d'un certain nombre d'images dans un seul fichier. Tk charge la première image mais vous pouvez spécifier différentes images en transmettant un paramètre d'index lors de la création de l'image. Par exemple:

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

Si vous chargez tous les cadres dans PhotoImages séparés, puis utilisez les événements du minuteur pour changer l'image affichée (label.configure (image = nextframe)). Le délai sur la minuterie vous permet de contrôler la vitesse d'animation. Il n'y a rien de prévu pour vous donner le nombre d'images dans l'image autre que le fait de ne pas créer une image une fois que vous avez dépassé le nombre d'images.

Lire Ajout d'images à une étiquette / bouton en ligne:
https://riptutorial.com/fr/tkinter/topic/9746/ajout-d-images-a-une-etiquette---bouton

Chapitre 3: Gestionnaires de géométrie Tkinter

Introduction

Il existe trois gestionnaires de géométrie pour positionner les widgets: `pack()` , `grid()` et `place()` .

Examples

pack()

Le gestionnaire de géométrie `pack()` organise les widgets en blocs avant de les placer dans le widget parent. Il utilise les options `fill` , `expand` et `side` .

Syntaxe

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

Remplir
Détermine si le widget conserve l'espace minimal requis ou occupe tout espace supplémentaire qui lui est alloué. Attributs: NONE (par défaut), X (remplissage horizontal), Y (remplissage vertical) ou BOTH (remplissage horizontal et vertical).

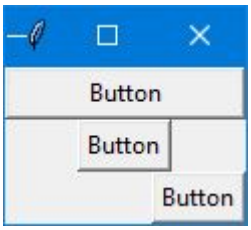
Développer
Lorsqu'il est défini sur YES, le widget se développe pour remplir tout espace non utilisé dans le parent du widget. Attributs: OUI, NON.

Côté
Détermine le côté du parent du widget auquel il est associé. Attributs: TOP (par défaut), BOTTOM, LEFT ou RIGHT.

Exemple

```
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()
```

Résultat



la grille()

Le gestionnaire de géométrie `grid()` organise les widgets dans une structure de type tableau dans le widget parent. Le widget maître est divisé en lignes et en colonnes et chaque partie de la table peut contenir un widget. Il utilise la `column` , `columnspan` , `ipadx` , `ipady` , `padx` , `pady` , `row` , `rowspan` et `sticky` .

Syntaxe

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

Colonne
La colonne dans laquelle placer le widget. La colonne par défaut est 0, qui est la colonne la plus à gauche.

Columnspan
Combien de widget de colonnes prend. La valeur par défaut est 1.

Ipadx
Combien de pixels pour remplir le widget horizontalement à l'intérieur des limites du widget.

Ipady
Combien de pixels pour créer un widget verticalement à l'intérieur des limites du widget.

Padx
Combien de pixels faut-il pour remplir le widget horizontalement en dehors des limites du widget.

Pady
Nombre de pixels à remplir à la verticale du widget en dehors des limites du widget.

Rangée
La ligne dans laquelle placer le widget. La ligne par défaut est 0, qui est la colonne la plus haute.

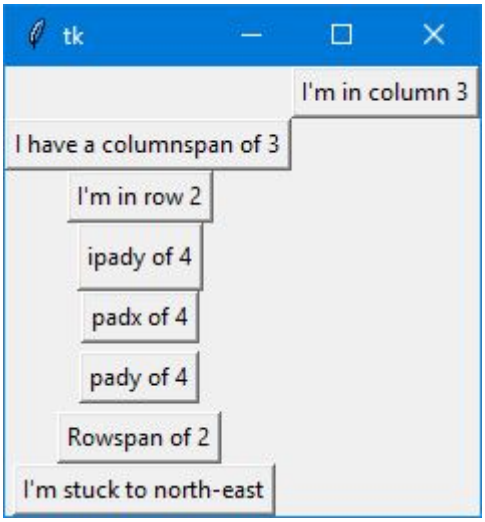
Rowspan
Combien de lignes le widget prend-il? La valeur par défaut est 1.

Gluant
Lorsque le widget est plus petit que la cellule, le `sticky` est utilisé pour indiquer les côtés et les coins de la cellule auxquels le widget adhère. La direction est définie par les directions de la boussole: N, E, S, W, NE, NW, SE et SW et zéro. Celles-ci peuvent être une concaténation de chaînes, par exemple, NESW fait que le widget occupe toute la surface de la cellule.

Exemple

```
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()
```

Résultat



endroit()

Le gestionnaire `place()` organise les widgets en les plaçant dans une position spécifique dans le widget parent. Ce gestionnaire de géométrie utilise les options `anchor` , `bordermode` , `height` , `width` , `relheight` , `relwidth` , `relx` , `rely` , `x` et `y` .

Ancre
Indique où le widget est ancré. Les options sont les directions de la boussole: N, E, S, W, NE,

NW, SE ou SW, qui se rapportent aux côtés et aux coins du widget parent. La valeur par défaut est NW (le coin supérieur gauche du widget)

Bordermode
Bordermode a deux options: `INSIDE` , qui indique que d'autres options se réfèrent à l'intérieur du parent, (ignorant les bordures du parent) et à l' `OUTSIDE` , ce qui est le contraire.

la taille
Spécifiez la hauteur d'un widget en pixels.

Largeur
Spécifiez la largeur d'un widget en pixels.

Racheté
Hauteur sous la forme d'un flottant compris entre 0,0 et 1,0, sous la forme d'une fraction de la hauteur du widget parent.

Relwidth
Largeur sous la forme d'un flottant compris entre 0,0 et 1,0, sous la forme d'une fraction de la largeur du widget parent.

Relx
Décalage horizontal sous forme de flottant compris entre 0,0 et 1,0, sous la forme d'une fraction de la largeur du widget parent.

Compter
Décalage vertical sous forme de flottant compris entre 0,0 et 1,0, sous forme de fraction de la hauteur du widget parent.

X
Décalage horizontal en pixels.

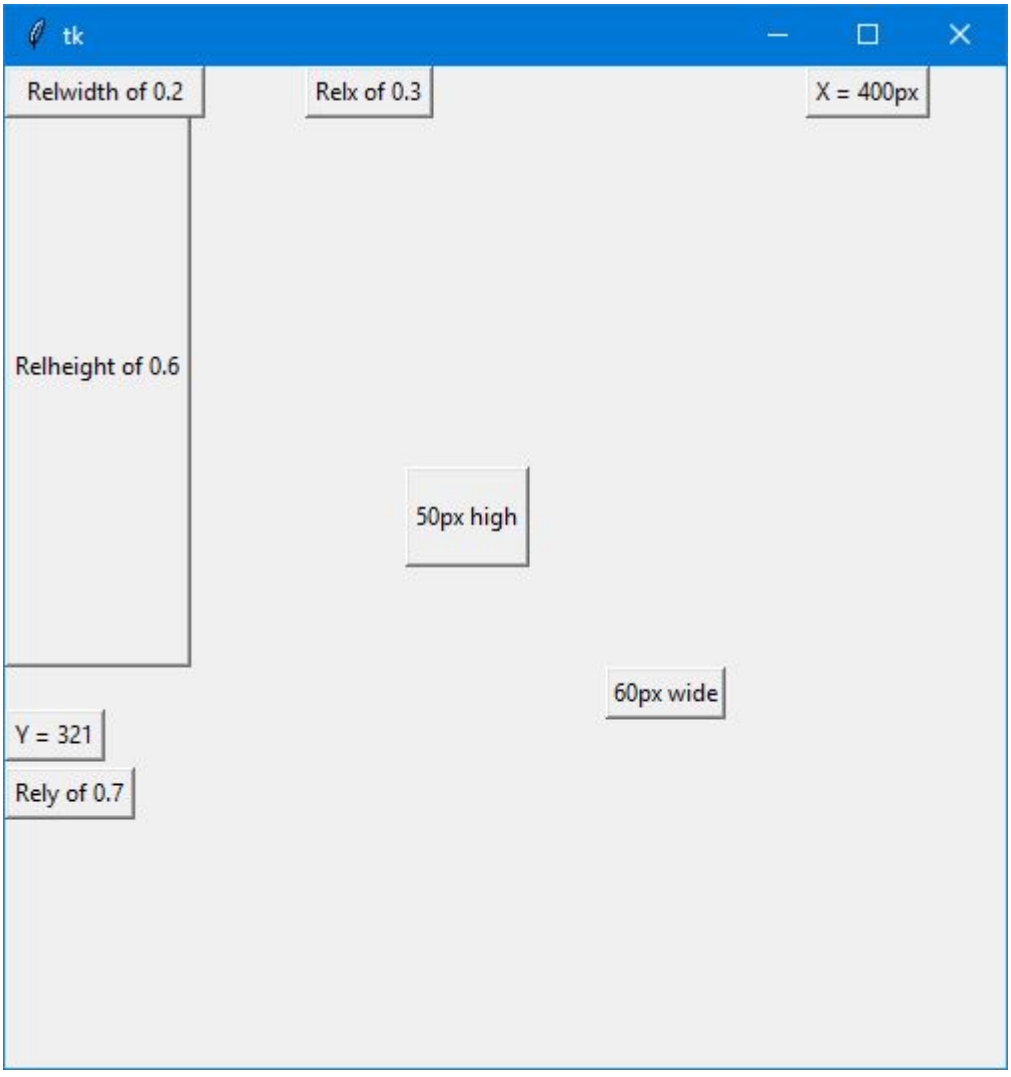
Y
Décalage vertical en pixels.

Exemple

```
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()
```

Résultat



Lire Gestionnaires de géométrie Tkinter en ligne:
https://riptutorial.com/fr/tkinter/topic/9620/gestionnaires-de-geometrie-tkinter

Chapitre 4: Le widget d'entrée Tkinter

Syntaxe

- entry = tk.Entry (*parent* , ** *kwargs*)
- entry.get ()
- entry.insert (index, "value")
- entry.delete (start_index, end_index)
- entry.bind (événement, rappel)

Paramètres

Paramètre	La description
parent	Les widgets tkinter existent dans une hiérarchie. À l'exception de la fenêtre racine, tous les widgets ont un parent. Certains didacticiels en ligne appellent ce "maître". Lorsque le widget est ajouté à l'écran avec le `pack` , le `place` ou la `grid` , il apparaîtra dans ce widget parent
largeur	La largeur spécifie la largeur *souhaitée* du widget en fonction d'une largeur de caractère moyenne. Pour les polices de largeur variable, cela est basé sur la largeur du caractère zéro (`0`). La valeur par défaut est 20. Notez que la largeur réelle peut être plus grande ou plus petite selon la manière dont elle est ajoutée à l'écran.

Remarques

Ces exemples supposent que tkinter a été importé avec `import tkinter as tk` (python 3) ou `import Tkinter as tk` (python 2).

Examples

Création d'un widget Entrée et définition d'une valeur par défaut

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

Obtenir la valeur d'un widget Entrée

La valeur d'un widget d'entrée peut être obtenue avec la méthode `get` du widget:

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

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

Vous pouvez éventuellement associer une instance de `StringVar` et extraire la valeur de `StringVar` plutôt que du widget:

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

Ajout de validation à un widget Entrée

Pour limiter les caractères pouvant être saisis dans un widget de saisie, seuls les nombres, par exemple, une commande de validation peuvent être ajoutés à l'entrée. Une commande validate est une fonction qui renvoie `True` si la modification est acceptée, `False` sinon. Cette fonction sera appelée chaque fois que le contenu de l'entrée est modifié. Différents arguments peuvent être passés à cette fonction, comme le type de modification (insertion, suppression), le texte inséré, ...

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

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

L'option `validate` détermine le type d'événement qui déclenche la validation, ici, c'est une frappe quelconque dans l'entrée. Le `'%S'` de l'option validatecommand signifie que le caractère inséré ou supprimé est passé en argument à la fonction `only_numbers` . La liste complète des possibilités se trouve ici .

Obtenir à partir du widget d'entrée

Lorsque vous utilisez la méthode .get (), tout ce qui se trouve dans le widget d'entrée sera converti en chaîne. Par exemple, quel que soit le type d'entrée (il peut s'agir d'un nombre ou d'une phrase), le résultat sera une chaîne. Si l'utilisateur tape 4, la sortie sera "4" comme dans une chaîne. Pour obtenir un int à partir d'un widget de saisie, commencez par appeler la méthode .get ().

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

Maintenant, nous convertissons cette chaîne en un int comme ça:

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

De même, si vous voulez gagner du temps, vous pouvez simplement faire:

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

Vous pouvez utiliser la méthode ci-dessus si vous ne voulez pas convertir str en int.

Lire Le widget d'entrée Tkinter en ligne: https://riptutorial.com/fr/tkinter/topic/4868/le-widget-d-entree-tkinter

Chapitre 5: Le widget Tkinter Radiobutton

Syntaxe

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

Paramètres

Paramètre	La description
parent	Les widgets tkinter existent dans une hiérarchie. À l'exception de la fenêtre racine, tous les widgets ont un parent. Certains didacticiels en ligne appellent ce "maître". Lorsque le widget est ajouté à l'écran avec le pack, le lieu ou la grille, il apparaîtra à l'intérieur de ce widget parent.
commander	fonction appelée chaque fois que l'utilisateur change l'état du bouton radio
indicatoron	1 ou True pour les boutons radio, 0 ou False pour les boutons
texte	Texte à afficher à côté du bouton radio.
valeur	Lorsque le bouton radio est sélectionné, la variable de contrôle associée est définie sur valeur.
variable	La variable de contrôle que le radiobutton partage avec l'autre radiobutton du groupe.

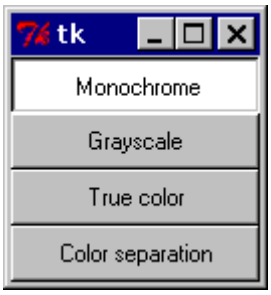
Remarques

Ces exemples supposent que tkinter a été importé avec `import tkinter as tk` (python 3) ou `import Tkinter as tk` (python 2).

Référence:

Pour transformer l'exemple ci-dessus en une «boîte à boutons» plutôt qu'en un ensemble de boutons radio, définissez l'option indicatorBon à 0. Dans ce cas, il n'y a pas d'indicateur de bouton radio distinct et le bouton sélectionné est dessiné comme

SUNKEN au lieu de RAISED:



- effbot

Examples

Voici un exemple de conversion de boutons radio en boutons:

```
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()
```

Créer un groupe de boutons radio

Un tel groupe est constitué de boutons radio partageant une variable de contrôle, de sorte qu'il ne soit pas sélectionné plus d'un.

```
# 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()
```

Lire Le widget Tkinter Radiobutton en ligne: https://riptutorial.com/fr/tkinter/topic/6338/le-widget-tkinter-radiobutton

Chapitre 6: Personnaliser les styles ttk

Introduction

Le style des nouveaux widgets ttk est l'un des aspects les plus puissants de ttk. Outre le fait qu'il s'agit d'une méthode de travail complètement différente de celle du paquet tk traditionnel, il permet d'effectuer une personnalisation importante de vos widgets.

Examples

Personnaliser une arborescence

En prenant Treeview: Exemple de base , on peut montrer comment personnaliser une arborescence de base.

Dans ce cas, nous créons un style "mystyle.Treeview" avec le code suivant (voir les commentaires pour comprendre ce que fait chaque ligne):

```
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
```

Ensuite, le widget est créé en donnant le style ci-dessus:

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

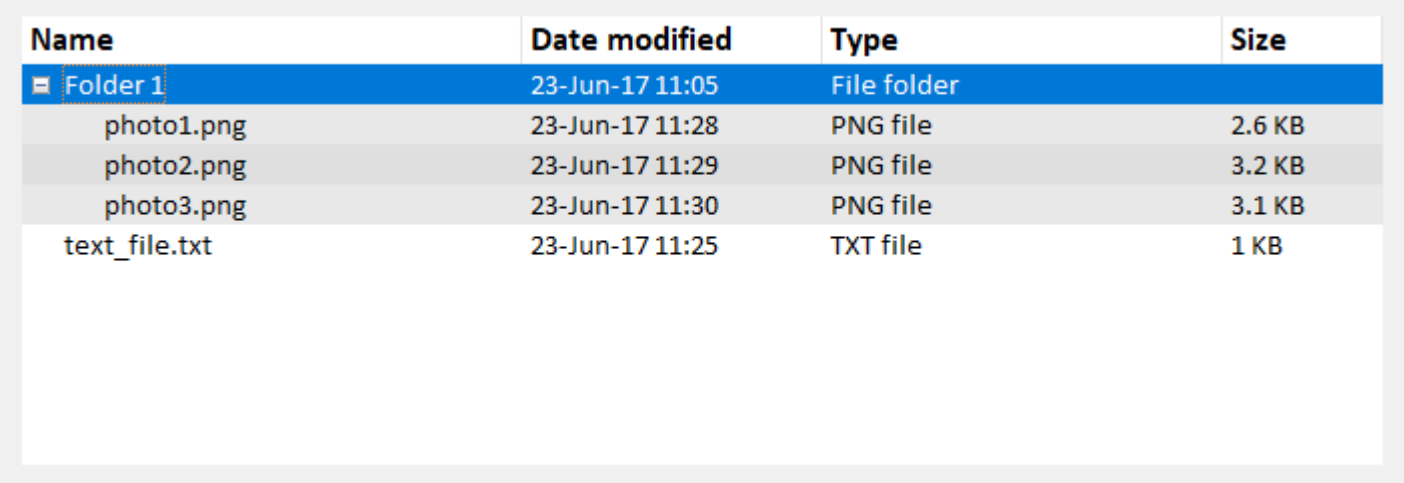
Si vous souhaitez avoir un format différent selon les lignes, vous pouvez utiliser des `tags` :

```
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',))
```

Par exemple, une couleur de fond peut être associée aux balises:

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

Le résultat est un treeview avec des polices modifiées à la fois sur le corps et les en-têtes, aucune bordure et différentes couleurs pour les lignes:



Remarque: Pour générer l'image ci-dessus, vous devez ajouter / modifier les lignes de code susmentionnées dans l'exemple Treeview: Exemple de base .

Lire Personnaliser les styles ttk en ligne: https://riptutorial.com/fr/tkinter/topic/10624/personnaliser-les-styles-ttk

Chapitre 7: Plusieurs fenêtres (widgets TopLevel)

Examples

Différence entre Tk et Toplevel

`Tk` est la racine absolue de l'application, c'est le premier widget qui doit être instancié et l'interface graphique s'arrête lorsqu'elle est détruite.

`Toplevel` est une fenêtre de l'application, la fermeture de la fenêtre détruira tous les widgets enfants placés dans cette fenêtre {1} mais ne fermera pas le programme.

```
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 votre programme python ne représente qu'une seule application (ce qui sera presque toujours le cas), vous ne devriez avoir qu'une seule instance `Tk` , mais vous pouvez créer autant de fenêtres `Toplevel` que vous le souhaitez.

```
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)`) est le parent d'un autre Toplevel (`B = Toplevel(A)`) alors la fermeture de la fenêtre A fermera également la fenêtre B.

organiser la pile de fenêtres (la méthode .lift)

Le cas le plus fondamental pour soulever une fenêtre particulière au-dessus des autres, il suffit d'appeler la méthode `.lift()` sur cette fenêtre (`Toplevel` ou `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()
```

Cependant, si cette fenêtre est détruite en essayant de la soulever, cela provoquera une erreur comme celle-ci:

```
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"
```

Souvent, lorsque nous essayons de placer une fenêtre particulière devant l'utilisateur mais qu'elle est fermée, une bonne alternative consiste à recréer cette fenêtre:

```
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 cette façon, la fonction `show_dialog` affichera la fenêtre de dialogue, qu'elle existe ou non, notez également que vous pouvez appeler `.winfo_exists()` pour vérifier si elle existe avant d'essayer de lever la fenêtre au lieu de l'enrouler pour `try:except` .

Il y a aussi la méthode `.lower()` qui fonctionne de la même manière que la méthode `.lift()` , sauf en abaissant la fenêtre dans la pile:

```
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()
```

Vous remarquerez que cela diminue même en dessous des autres applications, pour ne descendre que sous une certaine fenêtre, vous pouvez le passer à la méthode `.lower()` , de même cela peut aussi être fait avec la méthode `.lift()` un.

Lire Plusieurs fenêtres (widgets TopLevel) en ligne:
https://riptutorial.com/fr/tkinter/topic/6439/plusieurs-fenetres--widgets-toplevel-

Chapitre 8: Retarder une fonction

Syntaxe

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

Paramètres

Paramètre	La description
delay_ms	Temps (millisecondes) qui retarde l'appel à la fonction `callback`
rappeler	Fonction appelée après le `delay_ms` donné. Si ce paramètre n'est pas renseigné, `.after` agit de manière similaire à `time.sleep` (en millisecondes)

Remarques

La syntaxe suppose qu'un `widget` accepté par la méthode `.after` a déjà été créé (ie `widget=tk.Label(parent)`)

Examples

.après()

`.after(delay, callback=None)` est une méthode définie pour tous les widgets tkinter. Cette méthode appelle simplement la fonction `callback` après le `delay` donné en ms. Si aucune fonction n'est donnée, elle agit de manière similaire à `time.sleep` (mais en millisecondes au lieu de secondes)

Voici un exemple de la façon de créer une minuterie simple en utilisant `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()
```

Lire Retarder une fonction en ligne: https://riptutorial.com/fr/tkinter/topic/6724/retarder-une-fonction

Chapitre 9: Widgets de défilement

Introduction

Des barres de défilement peuvent être ajoutées aux widgets Listbox, Canvas et Text. De plus, les widgets d'entrée peuvent être défilés horizontalement. Pour pouvoir faire défiler d'autres types de widgets, vous devez les placer dans un widget Canvas ou Text.

Syntaxe

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

Paramètres

Paramètre	La description
parent	Les widgets tkinter existent dans une hiérarchie. À l'exception de la fenêtre racine, tous les widgets ont un parent. Certains didacticiels en ligne appellent ce "maître". Lorsque le widget est ajouté à l'écran avec le pack, le lieu ou la grille, il apparaîtra dans ce widget parent
Orient	Orientation de la barre de défilement, soit `"vertical"` (valeur par défaut) ou `"horizontal"`

Remarques

Ces exemples supposent que tkinter a été importé avec `import tkinter as tk` (python 3) ou `import Tkinter as tk` (python 2).

Examples

Connexion d'une barre de défilement verticale à un widget texte

La connexion entre le widget et la barre de défilement va dans les deux sens. La barre de défilement doit être agrandie verticalement pour avoir la même hauteur que le 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)
```

Faire défiler un widget Canvas horizontalement et verticalement

Le principe est essentiellement le même que pour le widget Texte, mais une disposition `Grid` est utilisée pour placer les barres de défilement autour du 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)
```

Contrairement au widget Texte, la zone déroulante du canevas n'est pas automatiquement mise à jour lorsque son contenu est modifié. Nous devons donc le définir et le mettre à jour manuellement à l'aide de l'argument `scrollregion` :

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

`canvas.bbox("all")` renvoie les coordonnées du rectangle correspondant à tout le contenu de la toile.

Faire défiler un groupe de widgets

Lorsqu'une fenêtre contient de nombreux widgets, ils ne sont peut-être pas tous visibles. Cependant, ni une fenêtre (instance Tk ou Toplevel) ni une image ne peuvent être défilées. Une solution pour faire défiler le contenu de la fenêtre consiste à placer tous les widgets dans un cadre, puis à intégrer cette image dans un canevas à l'aide de la méthode `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')
```

Lire Widgets de défilement en ligne: https://riptutorial.com/fr/tkinter/topic/8931/widgets-de-

defilement

Chapitre 10: Widgets Ttk

Introduction

Exemples des différents widgets ttk. Ttk dispose d'un total de 17 widgets, dont onze existaient déjà dans tkinter (tk).

L'utilisation du module ttk donne à votre application un aspect plus moderne et amélioré.

Syntaxe

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

Paramètres

Paramètre	La description
maîtriser	Les widgets tkinter existent dans une hiérarchie. À l'exception de la fenêtre racine, tous les widgets ont un parent (également appelé "master"). Lorsque le widget est ajouté à l'écran avec le pack, le lieu ou la grille, il apparaîtra dans ce widget parent

Remarques

Ces exemples supposent que tkinter a été importé avec `import tkinter as tk` (python 3) ou `import Tkinter as tk` (python 2).

On suppose également que ttk a été importé avec soit `from tkinter import ttk` (python 3), soit `import ttk` (python 2).

Examples

Treeview: exemple de base

Ce widget est utilisé pour afficher des éléments avec une hiérarchie. Par exemple, Windows Explorer peut être reproduit de cette manière. Quelques belles tables peuvent également être réalisées à l'aide du widget `treeview` .

Créer le widget

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

Définition des colonnes

Vous pouvez définir le nombre de colonnes, leur largeur et leur largeur minimale lorsque l'utilisateur essaie de l'étirer. En définissant `stretch=tk.NO` , l'utilisateur ne peut pas modifier la largeur de la colonne.

```
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)
```

Définition des rubriques

```
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)
```

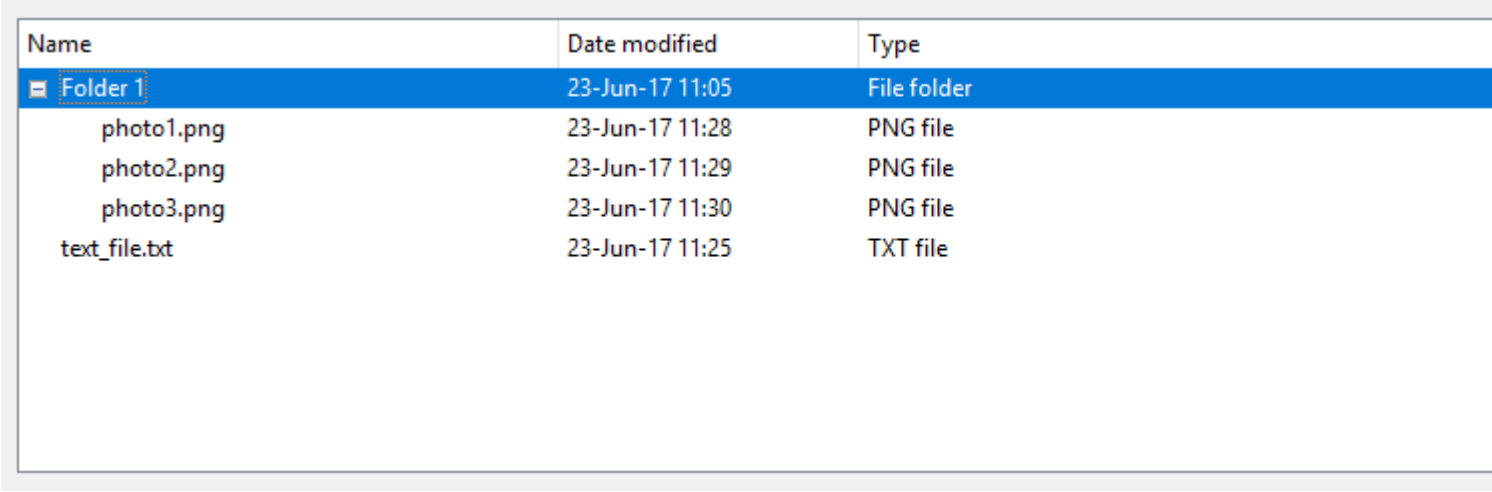
Insérer des lignes

```
# 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"))
```

Emballage

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

Sous Windows, la capture d'écran suivante peut être obtenue à partir de cet exemple.



Barre de progression

Le widget `ttk.progress` est utile pour traiter de longs calculs afin que l'utilisateur sache que le programme est en cours d'exécution. Après, un exemple de mise à jour d'une barre de progression toutes les 0,5 seconde est donné:

Fonction de mise à jour de la barre de progression

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

Définir la valeur maximale

```
maxValue=100
```

Créer la barre de progression

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

Le mode "déterminé" est utilisé lorsque la barre de progression est sous le contrôle du programme.

Valeurs initiales et maximales

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

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

Emuler la progression chaque 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
```

Lire Widgets Ttk en ligne: https://riptutorial.com/fr/tkinter/topic/10622/widgets-ttk

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec 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	Ajout d'images à une étiquette / bouton	Angrywasabi
3	Gestionnaires de géométrie Tkinter	Henry
4	Le widget d'entrée Tkinter	Angrywasabi, Bryan Oakley, double_j, j_4321
5	Le widget Tkinter Radiobutton	j_4321, nbro, Parviz Karimli
6	Personnaliser les styles ttk	David Duran
7	Plusieurs fenêtres (widgets TopLevel)	Tadhg McDonald-Jensen
8	Retarder une fonction	David Duran, Neil A., Tadhg McDonald-Jensen
9	Widgets de défilement	j_4321
10	Widgets Ttk	David Duran