Interface graphique

Zhentao Li

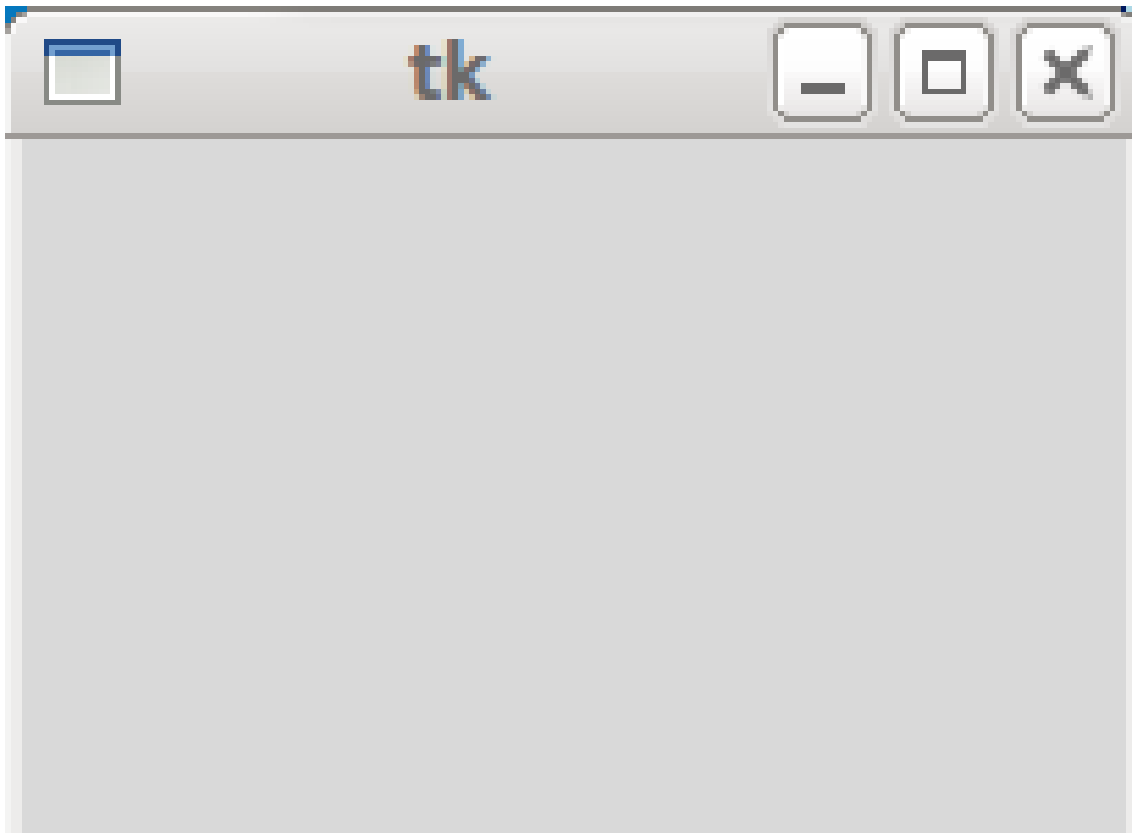
30 mars 2016

Module Tkinter (1)

```
import Tkinter as tk
```

Un exemple (en IPython):

```
In [1]: import Tkinter as tk
In [2]: racine = tk.Tk()
```



Module Tkinter (2)

```
In [3]: etiquette = tk.Label(racine, text='Bonjour')
In [4]: etiquette.pack()
```

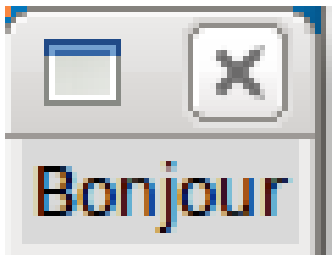


Figure 2:

Module Tkinter (3)

```
In [5]: bouton = tk.Button(racine, text='Quitter', comman
In [6]: bouton.pack()
```

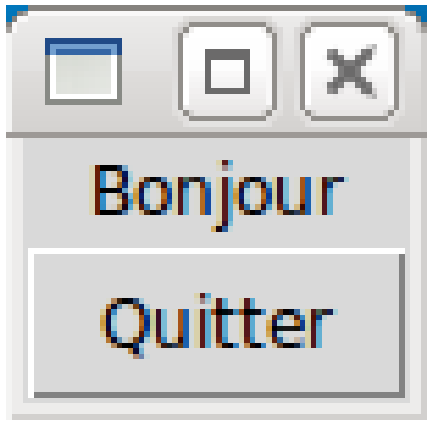


Figure 3:

Sans IPython, if faut ajouter

```
racine.mainloop()
```

Widgets Tkinter

Tkinter inclus des morceaux d'interface, appellés « widgets ».
Button: Button qui exécute un fonction lorsqu'on appui dessus.
Checkbutton: Case à cocher.
Entry: Champs d'entrée texte.
Menu: Un menu déroulant ou bien un menu pop up
Label: Étiquette contenant du texte (ou une image).
Radiobutton: Permet de choisir une valeur parmi plusieurs possibilités.
Tk: « Racine » de l'application graphique. C'est aussi la première fenêtre.
Toplevel: Une autre fenêtre qui n'est pas la « racine ».

Widgets un peu plus complexes

Canvas: Un espace pour disposer divers éléments dessinés (lignes, cercles, rectangles, etc).
Menubutton: Un bouton-menu, à utiliser pour les menus déroulants.
Listbox: Une liste de choix proposés à l'utilisateur.
Scrollbar: Barre de défilement
Text: Affichage de texte formatté. Permet aussi à l'utilisateur d'éditer le texte affiché.

Emplacement

Maintenant que nous avons vu quelques widgets de base, il faut spécifier où ils sont placé les uns par rapport aux autres.
L'emplacement voulue des widgets peut être décrite de trois façons différentes:

1. **Frame**, un contenant de d'autres widgets. `pack` place un widget dans un autre widget.
2. Utiliser `place` à la place de `pack` pour placer un élément à un endroit exacte.
3. Utiliser `grid` à la place de `pack` permet de disposer par coordonnées l'emplacement des widgets dans une grille.

Exemple Frame et pack (1)

```
import Tkinter as tk
racine = tk.Tk()
premier = tk.Label(racine, text='premier')
premier.pack(side='top')
deuxieme = tk.Label(racine, text='deuxième')
deuxieme.pack(side='top')
troisieme = tk.Frame(racine)
troisieme.pack(side='top')
```

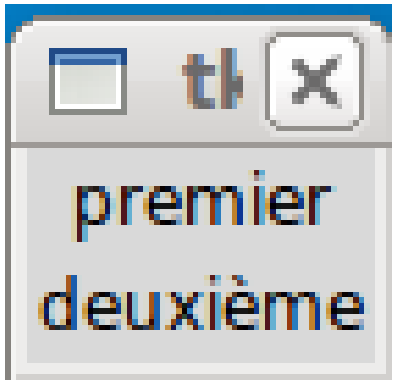


Figure 4:

Exemple Frame et pack (2)

```
gauche = tk.Label(troisieme, text='gauche')
gauche.pack(side='left')
milieu = tk.Label(troisieme, text='milieu')
milieu.pack(side='left')
droite = tk.Label(troisieme, text='droite')
droite.pack(side='left')
racine.mainloop()
```

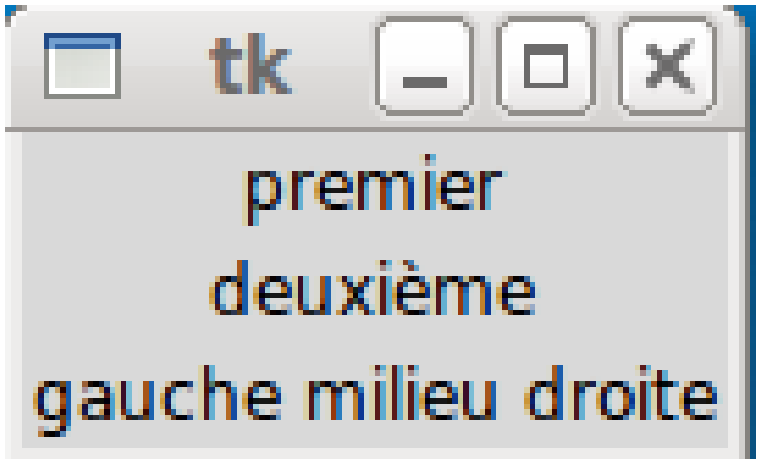


Figure 5:

pack

Remarque

Le premier paramètre donné lors de la création d'un widget est le widget qui devrait contenir celui-ci.

pack

Remarque

Le premier paramètre donné lors de la création d'un widget est le widget qui devrait contenir celui-ci.

- Les valeurs possible pour `side=` sont 'top', 'left', 'right', 'bottom'.

pack

Remarque

Le premier paramètre donné lors de la création d'un widget est le widget qui devrait contenir celui-ci.

- Les valeurs possible pour `side=` sont 'top', 'left', 'right', 'bottom'.
- Ici, les widgets contenus dans un `Frame` (ou `Tk` ou `Toplevel`) sont distribués uniformément. Pour voir les autres options possibles, lire le `help(tk.Label.pack)` ou les documents de référence.

pack

Remarque

Le premier paramètre donné lors de la création d'un widget est le widget qui devrait contenir celui-ci.

- Les valeurs possible pour `side=` sont 'top', 'left', 'right', 'bottom'.
- Ici, les widgets contenus dans un `Frame` (ou `Tk` ou `Toplevel`) sont distribués uniformément. Pour voir les autres options possibles, lire le `help(tk.Label.pack)` ou les documents de référence.

pack

Remarque

Le premier paramètre donné lors de la création d'un widget est le widget qui devrait contenir celui-ci.

- Les valeurs possible pour `side=` sont 'top', 'left', 'right', 'bottom'.
- Ici, les widgets contenus dans un `Frame` (ou `Tk` ou `Toplevel`) sont distribués uniformément. Pour voir les autres options possibles, lire le `help(tk.Label.pack)` ou les documents de référence.

Remarque (avancée)

D'avoir mis 'pack' comme méthode du widget contenu (plutôt que le widget contenant) est atypique de Python et est un vestige du langage Tcl/Tk. Le module 'Tkinter' de Python est une interface vers les fonctions de ce langage.

Exemple `place`

Si on ajoute ces deux lignes avant la fin du mainloop dans l'exemple précédent, on verrait le nouveau `Label` ajouté par dessus le reste.

```
flottant = tk.Label(racine, text='flottant')
flottant.place(x=10, y=20)
```

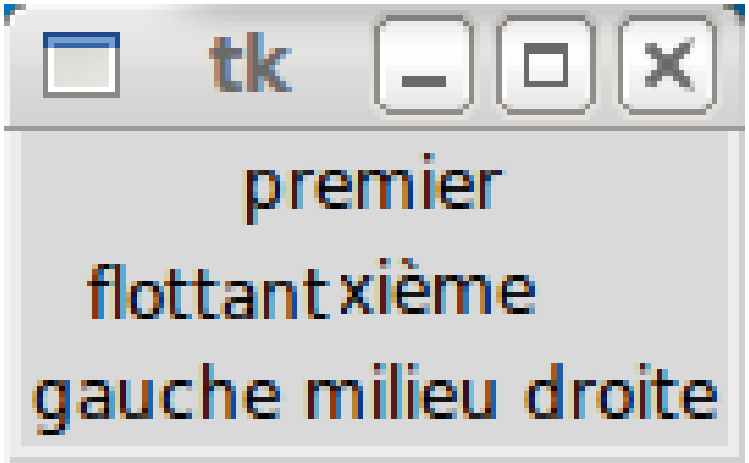


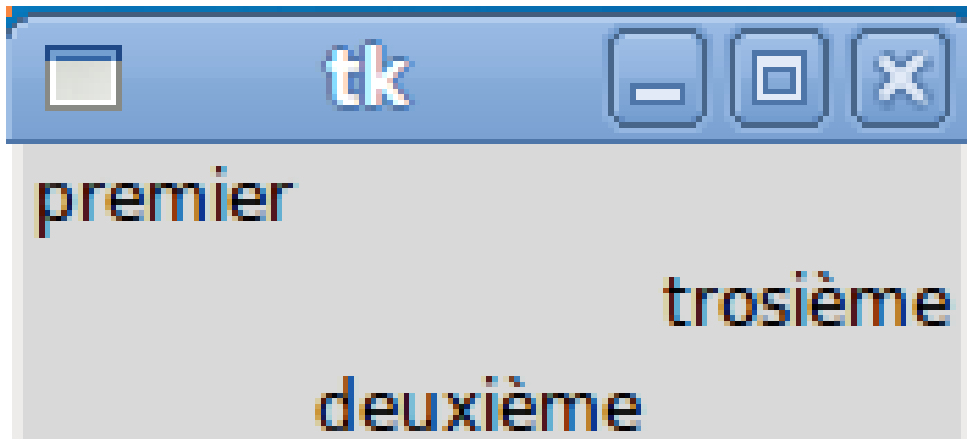
Figure 6:

Ici, nous appelons la méthode `place` **à la place de** `pack`.

Exemple grid

Nous pouvons aussi appeller la méthode grid à la place de pack.

```
import Tkinter as tk
racine = tk.Tk()
premier = tk.Label(racine, text='premier')
premier.grid(row=1, column=1)
deuxieme = tk.Label(racine, text='deuxième')
deuxieme.grid(row=3, column=2)
troisieme = tk.Label(racine, text='trosième')
troisieme.grid(row=2, column=3)
racine.mainloop()
```



Actions de l'utilisateur

Pour exécuter une fonction lorsque l'utilisateur appuie sur la touche Entrée sur un widget Entry, nous pouvons utiliser la

```
import Tkinter as tk

def touche_entree(widget):
    # Affiche le contenu entré
    print entry.get()

racine = tk.Tk()
entry = tk.Entry(racine)
entry.pack()
entry.bind('<Return>', touche_entree)
racine.mainloop()
```

bind

`bind` peut aussi associer une fonction aux clicks de la souris, etc.

Nom	Signification
<KeyPress>	Pression sur n'importe quelle touche
<KeyPress-a>	Pression sur la touche 'A' (minuscule)
<KeyPress-A>	Pression sur la touche 'A' (majuscule)
<Return>	Pression sur entrée
<Escape>	Touche d'échappement
<Up> <Down>	Pression sur les flèches
<Button-1>	Click souris gauche
<Button-2>	Click souris milieu (ou gauche et droite)
<Button-3>	Click souris droit
<ButtonRelease>	Fin de click gauche
<Motion>	Mouvement de la souris
<B1-Motion>	Mouvement de la souris avec click gauche
<Enter> <Leave>	Entrée et sortie souris d'un widget
<Configure>	Redimensionnement de la fenêtre
<Map> <Unmap>	Ouverture et iconification de la fenêtre

Idée du fonctionnement en général

- Lorsqu'une action est prise par l'utilisateur, l'idée est de récupérer la valeur des états et selon l'action, engendrer l'effet voulue.

Idée du fonctionnement en général

- Lorsqu'une action est prise par l'utilisateur, l'idée est de récupérer la valeur des états et selon l'action, engendrer l'effet voulue.
- Cette méthode de fonctionnement, où l'on réagit à des actions par des appels à des fonctions porte le nom de *programmation évènementielle*.

Idée du fonctionnement en général

- Lorsqu'une action est prise par l'utilisateur, l'idée est de récupérer la valeur des états et selon l'action, engendrer l'effet voulue.
- Cette méthode de fonctionnement, où l'on réagit à des actions par des appels à des fonctions porte le nom de *programmation évènementielle*.

Idée du fonctionnement en général

- Lorsqu'une action est prise par l'utilisateur, l'idée est de récupérer la valeur des états et selon l'action, engendrer l'effet voulue.
- Cette méthode de fonctionnement, où l'on réagit à des actions par des appels à des fonctions porte le nom de *programmation évènementielle*.

Question (avancée)

Comment est-ce que la programmation évènementielle est possible? Nous savons que les programmes sont pourtant exécutés ligne par ligne.

Références

Pour une liste plus exhaustive des fonctions de Tkinter et des paramètres que l'on peut leur passer.

- http://www.fil.univ-lille1.fr/~marvie/python/chapitre6.html
- http://python.developpez.com/cours/TutoSwinnen/?page=Chapitre8
- http://effbot.org/tkinterbook/
- http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html