Chapitre +

Initiation à Tkinter

Table des matières

La bibliothèque Tkinter (**T**ool **K**it **Inter**face) permet la gestion d'interface graphique en python.

1 La fenêtre principale

La création de l'objet principale de la bibliothèque Tkinter est la fenêtre à l'aide de la fonction `Tk`. Dans cette fenêtre nous pouvons placer des widget (windows gadget).
Les principaux widgets sont :

- les **Label** pour insérer un texte.
- les **Button** pour commander une action.
- les **Entry**, zones de saisie pour permettre l'entrée d'une donnée sous forme d'une chaine de caractères.
- les **Canvas** pour insérer des dessins ou des images.

Exemple :

```
from tkinter import *

ma_fenetre = Tk()

ma_premiere_phrase = Label(ma_fenetre,text="bonjour␣le␣monde")

ma_premiere_phrase.pack()

ma_fenetre.mainloop()
```

La fenêtre principale est la racine des objets qu'elle contient, c'est donc le premier argument dans la création de l'objet.

Exercice 1 :
Recopier le script :

```
from tkinter import *
ma_fenetre = Tk()
ma_fenetre.title('la␣fenetre')
ma_fenetre.geometry('400x300+400+400')
ma_fenetre['bg']="yellow"

ma_premiere_phrase = Label(ma_fenetre,text="bonjour␣le␣monde",fg = "blue")
ma_premiere_phrase.pack()

le_bouton = Button(ma_fenetre,text="Quitter", command = ma_fenetre.destroy)
le_bouton.pack(side="bottom")

ma_fenetre.mainloop()
```

Analyser les différentes commandes, et ajouter les commentaires.

1.1 Sous fenêtres avec Frame

Exercice 2 :

Recopier le script :

```
from tkinter import *

ma_fenetre = Tk() # on definit un objet de classe TK(instance de la classe )
ma_fenetre.title('tester le nom') # le titre
# ma_fenetre.geometry('400x300+400+400')# la taille et la position
ma_fenetre['bg']='bisque'



Frame1 = Frame(ma_fenetre,borderwidth=2,relief=GROOVE)
Frame1.pack(side=LEFT,padx=10,pady=10)

Frame2 = Frame(ma_fenetre,borderwidth=2,relief=GROOVE)
Frame2.pack(side=LEFT,padx=10,pady=10)

Frame4 = Frame(Frame1,bg = "blue", borderwidth=2,relief=GROOVE)
Frame4.pack(side=TOP,padx=10,pady=10)

Frame3 = Frame(Frame1,bg="white",borderwidth=2,relief=GROOVE)
Frame3.pack(side=RIGHT,padx=10,pady=10)

Label(Frame1,text="RDV dentiste samedi a 15h").pack(padx=10,pady=10)
Button(Frame1,text="Effacer",fg='navy',command=Frame1.destroy).pack(padx=10,
pady=10)

Label(Frame2,text="Reviser le controle d'info").pack(padx=10,pady=10)
Button(Frame2,text="Effacer",fg='navy',command=Frame2.destroy).pack(padx=10,
pady=10)

Label(Frame3,text="RDV dentiste a 10h",bg="white").pack(padx=10,pady=10)
Button(Frame3,text="Effacer",fg='navy',command=Frame3.destroy).pack(padx=10,
pady=10)

Label(Frame4,text="Les notes en sciences").pack(padx=10,pady=10)


ma_fenetre.mainloop()
```

1.2 Sous fenêtres avec Toplevel

Exercice 3 :

Recopier le script :

```
from tkinter import *

fenetre = Tk() # la fenetre principale
```

```
phrase = Label(fenetre,text="Bonjour")
phrase.pack()


bouton = Button(fenetre,text = "Quitter" , command = fenetre.destroy )
bouton.pack()

fenetre_2 = Toplevel(fenetre) #une premiere fenetre

phrase_2 = Label(fenetre_2,text = "Hello")
phrase_2.pack()
bouton_2 = Button(fenetre_2,text = "Quitter␣2" , command = fenetre_2.destroy )
bouton_2.pack()

fenetre_3 = Toplevel(fenetre)#une deuxieme fenetre

phrase_3 = Label(fenetre_3,text = "Hola")
phrase_3.pack()
bouton_3 = Button(fenetre_3,text = "Quitter␣3" , command = fenetre_3.destroy )
bouton_3.pack()


fenetre.mainloop()
```

1.3 Le canvas

Petit exemple :

```
from tkinter import *

fenetre = Tk()


# creation du canvas
canvas = Canvas(fenetre, width=500, height=500, bg="ivory")

# forme geometrique de base
rectangle = canvas.create_rectangle(50,60,150,160,fill="black")

cerle = canvas.create_oval(280,80,325,125,fill="pink")

polygone = canvas.create_polygon(80,250,200,200,120,350,fill = "blue")

for i in range(20) :
    canvas.create_line(5*i,0,0,100-5*i,fill="red")


#insertion d une image
photo=PhotoImage(file="image_python.GIF") # Ouverture de l image
img=canvas.create_image(250,250,image=photo)
```

```
#insertion de widgets
phrase = Label(text="bonjour")

bouton = Button(text = "Quitter" , command = fenetre.destroy )
petite_fenetre = canvas.create_window(50,150,window = phrase )
petite_fenetre_2 = canvas.create_window(50,200,window = bouton )


canvas.pack()
fenetre.mainloop()
```

2 Options et attributs de widgets

Voici une liste non-exhaustive des options :

- anchor : indique où le label doit être placé (N, NE, E, SE, S, SW, W, NW, ou CENTER(défaut))
- bg : précise la couleur de fond du label.
- bd : précise la taille de la bordure du label.
- height : Hauteur du label ;
- width : Largeur du label.
- padx : détermine l'espace à laisser à l'extérieur des bords est et ouest du label.
- pady : détermine l'espace à laisser à l'extérieur des bords nord et sud du label.
- relief : Style de la bordure du label (flat par défaut, raised, sunken, groove, ridge).
- state : précise le statut du label : NORMAL, ACTIVE ou DISABLED (désactivé).
- text : Texte à afficher dans le label.
- justify : Définit l'alignement de plusieurs lignes de texte dans le label (LEFT, RIGHT, CENTER(défaut)).
- image : affiche une image de la classe PhotoImage dans le label.

Remarque : La méthode configure(options) permet pour n'importe quel widget de modifier les propriétés des options .

Exemple :

```
from tkinter import *

fenetre = Tk()

phrase = Label(fenetre,text="bonjour" , bg = "blue" , bd = 6 , height = 5 ,\
               width = 15 , padx = 5 , pady = 7 , relief = "groove")
phrase.pack()

fenetre.mainloop()
```

3 Placement dans la fenêtre

3.1 Méthode pack

Elle permet d'empiler les objets les uns à la suite des autres, par défaut les uns en dessous des autres.

- Avec l'option side en choisissant RIGHT, LEFT, BOTTOM ou TOP(par défaut), on place l'objet à l'endroit indiqué.
- Avec l'option expand,en choisissant True ou False(par défaut), l'objet occupe tout l'espace si l'option est vraie.
- Avec l'option padx, on détermine l'espace à laisser à l'extérieur des bords est et ouest.
- Avec l'option ipadx, on détermine l'espace à laisser à l'intérieur des bords est et ouest. (Les options pady et ipady ont les mêmes rôles avec les bords nord et sud.)

3.2 Méthode grid

Elle place les objets dans un tableau dont chaque case peut recevoir un objet, il suffit d'indiquer la ligne et la colonne.
Les options principales sont column, row, rowspan, columnspan et sticky.

- Avec l'option column, on indique la colonne dans laquelle sera placé l'objet.
- Avec l'option row, on indique la ligne à laquelle sera placé l'objet.
- Avec les options rowspan et columnspan, on indique le nombre de lignes et de colonnes que doit occuper l'objet.
- Avec l'option sticky, on a la possibilité d'aligner l'objet sur un ou plusieurs bords : "N", "S", "W", "E", "N+S" ou "E+W"

3.3 Méthode place

Elle place les objets à une position définie par des coordonnées.
Pour l'utiliser, on écrit : N.place(x=...,y=...) où N est le nom du widget.

4 Gestion des évènements

4.1 Méthode bind

Exemple :

```
from tkinter import *
fenetre = Tk()
# creation du canvas
canvas = Canvas(fenetre, width=500, height=500, bg="ivory")

rectangle = canvas.create_rectangle(50,60,150,160,fill="black")

def bouge(event):
```

```
    touche = event.keysym
    if touche == "Up":
        canvas.move(rectangle,0,-1)
    elif touche == "Down":
        canvas.move(rectangle,0,1)
    elif touche == "Right":
        canvas.move(rectangle,1,0)
    elif touche == "Left":
        canvas.move(rectangle,-1,0)

canvas.focus_set()
canvas.bind("<Key>", bouge)

canvas.pack()
fenetre.mainloop()
```

4.2 Gestion des mouvements

Exemple :

```
from tkinter import *
def deplacement():
    global dx, dy
    if dx > 0 :
        dx +=0.01
    else :
        dx -= 0.01
    if dy > 0 :
        dy +=0.01
    else :
        dy -=0.01
    if canvas.coords(balle1)[3]>=400 :
        dy = - dy
    if canvas.coords(balle1)[1]<=0 :
        dy = - dy
    if canvas.coords(balle1)[0]<=0 :
        dx = - dx
    if canvas.coords(balle1)[2]>=500 :
        dx = - dx
    #On deplace la balle :
    canvas.move(balle1,dx,dy)

    affiche.configure(text = "␣la␣vitesse␣en␣x␣:␣" + str(dx) + \
            "la␣vitesse␣en␣y␣:␣" + str(dy))
    #On repete cette fonction
    tk.after(20,deplacement)

#Deplacement de la balle au depart:
dx=1
dy=5
 #On cree une fenetre et un canevas:
```

```
tk = Tk()
canvas = Canvas(tk,width = 500, height = 400 , bd=0, bg="white")
canvas.pack(padx=10,pady=10)

affiche = Label(tk , text = "␣la␣vitesse␣en␣x␣:␣" + str(dx) + \
            "la␣vitesse␣en␣y␣:␣" + str(dy) )
affiche.pack()
#Creation  d'un bouton "Quitter":
Bouton_Quitter=Button(tk, text ='Quitter', command = tk.destroy)
#On ajoute l'affichage du bouton dans la fenetre tk:
Bouton_Quitter.pack()

#On cree une balle:
balle1 = canvas.create_oval(10,10,30,30,fill='red')

deplacement()
tk.mainloop()
```

4.3 Gestion de la souris

Exemple :

```
from tkinter import *
fenetre = Tk()
canvas = Canvas(fenetre, width=500, height=500, bg="ivory")
nb_point = 0
abs_de_depart = 0
ord_de_depart = 0
def dessin(evenement) :
    global nb_point, abs_de_depart , ord_de_depart
    if nb_point == 1 :
        abs_arrivee = evenement.x
        ord_arrivee = evenement.y
        canvas.create_line(abs_de_depart,ord_de_depart,abs_arrivee \
                           ,ord_arrivee ,fill="red")
    else :
        abs_de_depart = evenement.x
        ord_de_depart = evenement.y
    nb_point = (nb_point  +1) % 2

def efface() :
    global nb_point, abs_de_depart , ord_de_depart
    nb_point = 0
    abs_de_depart = 0
    ord_de_depart = 0
    canvas.delete("all")

canvas.pack()
fenetre.bind("<Button-1>" , dessin )
bouton_nettoyage = Button(fenetre, text="Effacer" , command = efface )
bouton_nettoyage.pack()
fenetre.mainloop()
```