Formation Python

PAR ALAIN CARIOU, NOVEMBRE 2023

I – Introduction aux GUI avec Tkinter

Qu’est-ce que Tkinter ?

- **Tkinter** est une bibliothèque graphique multiplate-forme pour Python.
- Il s’agit d’une adaptation de la bibliothèque graphique Tk utilisée par le langage de script TCL.
- Elle est disponible sous licence BSD.
- Il existe d’autres librairies graphiques sous Python tel que **Qt for Python**, **PyQT**, **kivy** ou encore **Pygame**.

Installer Tkinter

- Pour installer, vous pouvez simplement écrire dans un terminal :
    - ***pip install tk***

- Pensez à créer un nouvel environnement virtuel pour votre projet avant d'y installer Tkinter. ;-)

« Hello World » avec Tkinter - 1

- Voici un exemple de « Hello World » avec Tkinter :

```
from tkinter import *

windows = Tk()

label = Label(windows, text="Hello World")
label.pack()

windows.mainloop()
```

« Hello World » avec Tkinter - 2

- On doit commencer par initialiser une fenêtre en appelant la méthode **Tk()**.
- Ensuite on déclare un **Label** qui sera par la suite positionné dans la fenêtre via sa méthode ***pack()***.
- Le label prend un paramètre un objet auquel il est rattaché, ici la fenêtre, et le texte que l'on veut afficher. Il peut aussi prendre tout un ensemble d'autres paramètres.
- Puis on affiche la fenêtre via sa méthode ***mainloop()***.

II – La géométrie dans Tkinter

Le placement des éléments

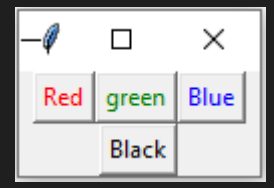
- Tkinter permet de placer des **widgets** dans une fenêtre. Cela peut être des boutons, des labels, des inputs, des barres de progression, etc.

- Tkinter propose trois méthodes de gestion de la géométrie afin de placer les widgets :
    - la méthode **pack()** qui permet de placer les widgets sous forme de blocs.
    - la méthode **grid()** qui permet de placer les widgets sous forme de tables.
    - la méthode **place()** qui permet de placer les widgets selon des positions spécifiques.

La méthode pack()

- **pack()** prendre 3 options :
  - **expand** : *True* ou *False* ; indique si l'élément prend toute la place.
  - **fill** : *NONE*, *X*, *Y*, *BOTH* ; détermine si le widget occupe l'espace additionnel.
  - **side** : *TOP*, *BOTTOM*, *LEFT*, *RIGHT* ; détermine contre quel bord du widget parent se trouve le widget.

```
from tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()

bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text="Red", fg="red")
redbutton.pack( side = LEFT)

greenbutton = Button(frame, text="green", fg="green")
greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text="Blue", fg="blue")
bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text="Black", fg="black")
blackbutton.pack( side = BOTTOM)

root.mainloop()
```

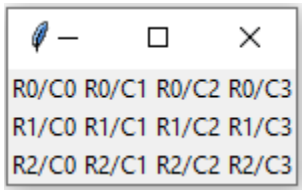


La méthode grid() - 1

- **grid()** permet de gérer le placement des widgets selon une structure en forme de tables à travers les options suivantes :
  - **column** : la colonne où mettre le widget.
  - **columnspan** : le nombre de colonnes occupées par le widget.
  - **ipadx**, **ipady** : le padding intérieur du widget en nombre de pixels.
  - **padx**, **pady** : le padding extérieur du widget en nombre de pixels.
  - **row** : la ligne où se trouve le widget.
  - **rowspan** : le nombre de colonnes occupées par le widget.
  - **sticky** : comment développer le widget si la cellule dans laquelle il se trouve est plus grande que lui. Par défaut, il est centré dans la cellule.

La méthode grid() - 2

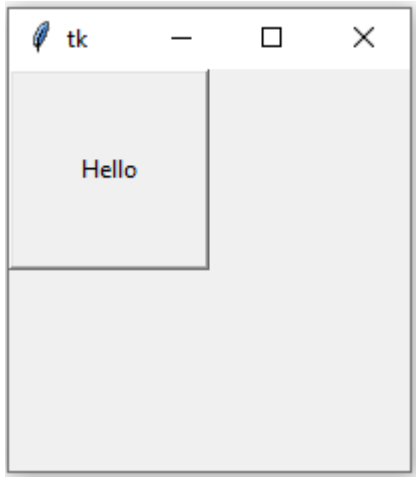
```
import tkinter

root = tkinter.Tk(  )
for r in range(3):
    for c in range(4):
        tkinter.Label(root, text='R%s/C%s'%(r,c), borderwidth=1).grid(row=r, column=c)
root.mainloop()
```



La méthode place() - 1

- place() permet de placer des widgets dans la fenêtre selon une position spécifique grâce aux options suivantes :
  - **anchor** : la direction dans laquelle le widget est ancré (N, E, S, W, NE, NW, SE, ou SW).
  - **bordermode** : *INSIDE*, *OUTSIDE* ; indique si les bordures sont intérieures ou extérieurs.
  - **height**, **width** : hauteur et largeur en pixels.
  - **relheight**, **relwidth** : hauteur et largeur selon un pourcentage de la hauteur et largeur du widget.
  - **relx**, **rely** : décalage horizontal and vertical selon un pourcentage de la hauteur et de la largeur du widget.
  - **x**, **y** : décalage horizontal et vertical en pixels.

La méthode place()




```
import tkinter
from tkinter import OUTSIDE, messagebox

top = tkinter.Tk()

def helloCallBack():
   messagebox.showinfo( "Hello Python", "Hello World !")

B = tkinter.Button(top, text="Hello", command=helloCallBack)

B.pack()
B.place(bordermode=OUTSIDE, height=100, width=100)
top.mainloop()
```

III – Les widgets Tkinter

Les labels

- Les **labels** sont des espaces prévus pour écrire du texte.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-label/*

```
label = Label(window, text="Hello World")
label.pack()
```

Les boutons

- Tkinter propose des **boutons** classiques permettant d'effectuer une action prédéfinie.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-button/*

```
from tkinter import *

def clickBtn():
    print("ok")

window = Tk()

label = Label(window, text="Hello World")
label.pack()

btn1 = Button(window, text="OK", command=clickBtn)
btn1.pack()
btn2 = Button(window, text="Fermer", command=window.quit)
btn2.pack()

window.mainloop()
```

Les frames

- Les **frames** (cadres) sont des conteneurs qui permettent de séparer des éléments.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-frame/*

```
window = Tk()

window['bg']='white'

Frame1 = Frame(window, borderwidth=2, relief=GROOVE)
Frame1.pack(side=LEFT, padx=30, pady=30)

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

Frame3 = Frame(Frame2, bg="white", borderwidth=2, relief=GROOVE)
Frame3.pack(side=RIGHT, padx=5, pady=5)

Label(Frame1, text="Frame 1").pack(padx=10, pady=10)
Label(Frame2, text="Frame 2").pack(padx=10, pady=10)
Label(Frame3, text="Frame 3",bg="white").pack(padx=10, pady=10)

window.mainloop()
```

Les inputs

- Les **inputs** sont gérés par la méthode **Entry**. Ils permettent de récupérer des données entrées par l'utilisateur.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-entry/*

```
from tkinter import *

def displayData():
    print(entry_user.get())
    print(entry_pwd.get())

window = Tk()
window.geometry("200x150")

entry_user = Entry(window, width = 20)
entry_user.insert(0, 'Username')
entry_user.pack(padx = 5, pady = 5)

entry_pwd = Entry(window, width = 15)
entry_pwd.insert(0, 'password')
entry_pwd.pack(padx = 5, pady = 5)

Button = Button(window, text = "Submit", command = displayData)
Button.pack(padx = 5, pady = 5)

window.mainloop()
```

Les cases à cocher

- Les **checkbox** proposent à l'utilisateur de cocher une option.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-check-button/*

```
from tkinter import *

window = Tk()

var1 = IntVar()
var2 = IntVar()

btn1 = Checkbutton(window, text="Donner des madeleines", variable=var1)
btn1.pack()
btn2 = Checkbutton(window, text="Donner d'autres madeleines", variable=var2)
btn2.pack()

window.mainloop()
```

Les listes

- Les **listes déroulantes** permettent de choisir une élément parmi une liste. Elles sont appelées ici **Combobox**.

- Doc : *https://coderslegacy.com/python/python-gui/python-tkinter-combobox/*

```
from tkinter import Tk, Frame, ttk

window = Tk()

frame = Frame(window)
frame.pack()

comboList = ["1", "2", "3", "La réponse D", "Obiwan Kenobi"]

combo = ttk.Combobox(frame, values = comboList)
combo.set("Choisissez une réponse")
combo.pack(padx=5, pady=5)

window.mainloop()
```

Les images

- Tkinter permet d'afficher des images à travers sa classe **PhotoImage**.

- Bien qu'efficace, cette classe ne gère pas beaucoup de formats d'images et, dans ce cas, il faudrait utiliser un autre module tel que **ImageTk** de la bibliothèque **Pillow**.

- PhotoImage gère uniquement les formats suivant : GIF, PGM, PPM et PNG.

Les images avec PhotoImage

- L'image est chargé à travers la classe **PhotoImage** puis elle peut être ensuite affichée à travers un label, un bouton, ou encore un canva.

- Doc : *https://coderslegacy.com/python/tkinter-photoimage/*

```
from tkinter import PhotoImage, Tk, Label

window = Tk()

img = PhotoImage(file="marmotte.png")
label = Label(window, image = img)
label.pack()

window.mainloop()
```

Exercice 1 : voyage, voyage

- Créez une classe Voyage contenant les propriétés nom, pays, ville, description, image, dateDebut, dateFin et prix ; ainsi qu'une méthode afficher() permettant de visualiser les données de cette classe.

- Concevez ensuite une interface graphique avec Tkinter pour afficher votre voyage.

TP – une interface pour nos personnages

- Reprenez le TP « jeu de rôle » et essayer de développer une interface graphique permettant de visualiser les personnages que vous avez créé.

- Ajouter ensuite une fenêtre permettant de créer de nouveaux personnages à partir de votre application.