

Tkinter.

Tkinter est une interface graphique dite de bureau (Word, courriel, jeu sans animation (Solitaire), etc...)

Chaque objet de Tkinter (Button, Text, Menu, etc...) est appelé widget.

A chaque fois que l'on crée un widget, il n'est pas affiché. On doit les afficher grâce à 1 des 3 méthodes suivantes et 1 seule seulement :

- Méthode `.pack()`: Place le widget dans l'espace sans utiliser de coordonnées. On a la possibilité de placer légèrement notre widget en mettant l'argument `size` et en lui donnant une valeur suivante :

• `tkinter.CENTER`

• `tkinter.BOTTOM`

• `tkinter.LEFT`

• `tkinter.TOP`

• `tkinter.RIGHT`

`mon_widget.pack(size=tkinter.CENTER)`

Pemet de mettre mon widget au centre de son espace.

- Méthode `.grid()`: Place le widget dans l'espace en utilisant un semblant de coordonnées. Prend 2 arguments, `colspan` et `ligne`, où l'on placera notre widget. Si la colonne est égal à la ligne qui est égal à 0. Le widget sera à l'ant à gauche. Le nombre de colonne et de ligne n'est pas défini, vous redimensionner votre écran à chaque fois que vous rajoutez une ligne / colonne. grid prend également 2 autres argument (optionnel, cette fois), `padx` et `pady`, correspondant à l'espace entre 2 widget pour.

x et y respectivement.

`mon_widget.grid(column=1, row=2, padx=2)`

Permet de mettre mon widget dans la 2^e colonne et 3 ligne avec un espace de 2 (pixel, il me semble) pour les x avec les autres widget.

- Méthode `.place()`: Place le widget dans l'espace en utilisant des coordonnées fixes. On doit lui fournir 2 argument correspondant aux coordonnées. On peut utiliser l'argument anchot avec N, E, S, W, NE, NW, SE, SW (point cardinaux)

`mon_widget.place(x=200, y=150)`

Permet de placer mon widget au point de coordonnée (200,150)

Les fenêtres

les fenêtres sont des objets Tkinter comme les autres, il faut les créer puis afficher.

Il existe 2 manières de créer les fenêtres :

- `Tkinter.Tk()`: On préfère l'utiliser 1 fois pour la fenêtre principale (au début du programme principal).

- `Tkinter.Toplevel()`: On utilise pour les autres fenêtres

On peut ajouter un titre à la fenêtre.

`fenetre = Tkinter.Tk()`

`fenetre.title("Mon titre")`

On peut définir la taille de la fenêtre.

fenetre.geometry("350x200")

crée une fenêtre avec 350 pixels en x et 200 en y.

On peut également modifier la couleur de fond.

fenetre.config(bg="green")

fenetre.config(bg="#87CEEB")

On peut mettre un icône de fenêtre (à côté du titre)

fenetre.iconbitmap("logo.ico")

On affiche la fenêtre (à la dernière ligne du programme).

fenetre.mainloop()

les Widgets

Les widgets ont des arguments obligatoire mais également beaucoup d'arguments optionnelle. C'est pourquoi je ne mettrai pas tous et les obligatoires seront sur lignes.

Widget de Texte (Label)

Texte = tkinter.Label(master=fenetre, text="Bongou",
bg="green", font=("Int Frc", 20))

Texte.pack()

Le premier paramètre correspond à l'espace où le widget sera afficher.

le second paramètre, `text` ou `textvariable`, selon si l'on souhaite avoir un texte constant ou un texte pouvant varier. Si on veut faire varier notre texte, on doit mettre une variable initialisée avec la méthode `StringVar()`.

Variable = `tkinter.StringVar()`

`Texte2 = tkinter.Label(fenetre, textvariable=Variable)`

Le `.get()` permet de récupérer la valeur d'une `StringVar()`. Le `.set()` permet lui de modifier cette valeur.

Mon troisième argument, `bg`, modifie la couleur du fond d'écran. On peut aussi utiliser `fg` pour la couleur du texte, ou `bd` pour modifier la largeur de bordure en pixels (2 par défaut)

Mon dernier argument, `font`, permet de personnaliser la police et la taille d'écriture.

On peut également mettre une image ou un objet bitmap à la place du texte en utilisant les arguments `bitmap` ou `image`. Nous pouvons justifier le texte à gauche avec `LEFT`, au centre avec `CENTER` ou à droite avec `RIGHT`.

On peut également souligner un caractère avec `underline` égal au numéro du caractère à souligner, ou encapsuler les lignes de texte dans une longueur avec `wraplength`.

Enfin on peut modifier le relief de la zone de texte avec relief prenant les valeurs `SUNKEN`, `RAISED`, `GROOVE` et `RIDGE`

Widget de Bouton (Button)

Bouton = `tkinter.Button(fenetre, image = image, state = ACTIVE, activebackground = TRUE, command : fonction)`

On remarque qu'un bouton se construit comme un Texte. On peut utiliser les même argument. On peut également rajouter `activebackground` et `activeforeground` qui change la couleur, lorsque on clique dessus de l'arrière plan et du premier plan (texte). On a aussi `state` qui permet de griser le bouton, si il est sur DISABLES, comme si on ne pouvait pas l'utiliser.

Une dernière commande, très apprécié dans les Button est l'argument `command`. Il permet de mettre une fonction qui sera exécuter lorsque le bouton est cliqué. On peut également mettre une méthode à la place d'une fonction (`def list, fenetre.destroy, etc...`). Il y a tout de même une petit subtilité, on ne met pas de parenthèse; sinon notre fonction/méthode sera exécuter à la création du bouton! Si nous avons besoin de mettre des argument à notre fonction, on utilise `lambda` qui permettra d'exécuter votre fonction qu'à l'appel du bouton.

Bouton = `tkinter.Button(fenetre, command = lambda : fonction(arg1, arg2))`

Pensons également que les image ne peuvent pas être afficher directement dans Tkinter, nous devons les convertir grâce à `PhotoImage`:

```
image = tkinter.PhotoImage(file = "img/cute5-P.gif")
```

Les Canvas

les canvas sont des objet complexe, j'ai pris pour habitude de les considerer comme un complement Turtle car on les utilise beaucoup pour le dessin. Mais il sont plus puissant que ça. En effet, nous pouvons également les comparer à une fenêtre Tkinter car il peuvent contenir tout les widget. Attention tout de même, pour que un widget soit dans un canvas, il faut modifier l'argument master, jusqu'ici fenêtre en mon_canvas.

```
mon_canvas = tkinter.Canvas(fenetre, bg = "green",  
                             height = 100, width = 200,  
                             highlightthickness = 0)
```

```
mon_canvas.grid(column = 1, row = 0)
```

Creation et affichage d'un canvas de 100x200 pixel et une bordure de 0 pixel.

Mettre un button dans ce canvas :

```
Bouton = tkinter.Button(mon_canvas, bg = "yellow")
```

```
Bouton.pack()
```

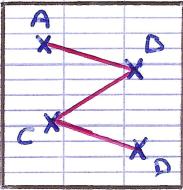
Pour Dessiner dans un canvas, il faut connaître les differents items.

- `create_line` pour dessiner une ligne.

```
mon_canvas.create_line((50, 60), (150, 60))
```

```
can.create_line(coord_depart, coord_arriver)
```

On peut également faire `.create_line(A, B, C, D)`

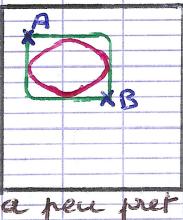


Il existe aussi les arguments `width` pour l'épaisseur du trait et `fill` pour la couleur du trait et `arrow` pour créer une flèche (`arrow="last"`)

- `create_rectangle` pour dessiner un rectangle (ou carré)
`mon_canvas.create_rectangle((280, 40), (400, 200), fill = "orange")`
`can.create_rectangle(coord_debut, coord_arriver, option)`

L'argument `fill` ici colorise l'intérieur du rectangle, pour colorier la bordure on utilisera `outline`. Pour la supprimer, on mettra une chaîne de caractère vide à la place d'une couleur (dans `outline`)

- `create_oval` pour dessiner un disque, un cercle.
Pour Tkinter, un cercle est inscrit dans un rectangle



C'est pour cela qu'un cercle se crée strictement pareille qu'un rectangle.

`mon_canvas.create_oval((280, 40), (400, 200), outline = "", fill = "red")`

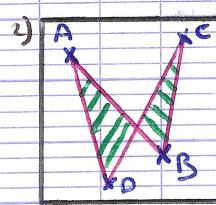
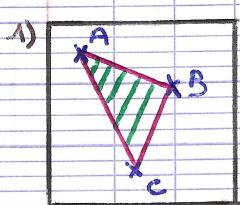
- `create_tktext` dessine du texte. Attention le texte est bien dessiner, ce n'est pas un widget (`label`)
`mon_canvas.create_tktext((230, 40), anchor = tkinter.W, text = "coucou", font = "Arial 30 bold")`

Le premier argument est la position du texte, le deuxième (`anchor`) son alignement, son troisième (`text`) est le texte à afficher et enfin `font` est la police d'écriture.

N	est aussi
W	coucou
S	NW, NE SW, SE C

il existe aussi l'option angle permettant d'incliner le tessez.

- `create_polygon` pour créer des polygones.



1) `can.create_polygon(A, B, C, fill="green")`

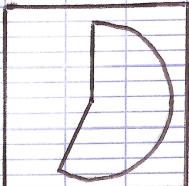
2) `can.create_polygon(A, B, C, D, outline="red")`

- `create_arc` pour créer des arcs de cercle.

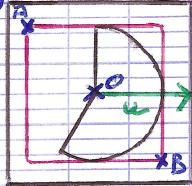
Il existe 3 style d'arc :

1) non préciser

`mon_canvas.create_arc((280, 40), (400, 200), extent=210, start=-120)`



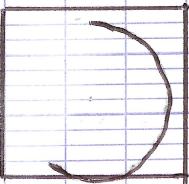
option `extent` est la totalité de l'angle (OA, OB)
angle en degrés (w, OA)



2) arc

`mon_canvas.create_arc((280, 40), (400, 200), extent=210, start=-120,`

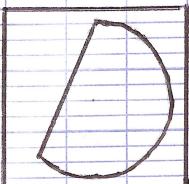
`style=tkinter.ARC)`



3) chord

`mon_canvas.create_arc((280, 40), (400, 200), extent=210, start=-120`

`style=tkinter.CHORD)`



Les cases à cocher (Radiobutton et checkbutton)

L'unique différence entre les checkbuttons et les Radiobutton est le nombre de case cochable. En effet, alors que les Radiobutton permettent de cocher qu'une seule case, les checkbutton permettent d'en cocher en 0 et n. Malgré cela leur syntaxe est exactement la même.

Ma_Var = tkinter.StringVar()

Bouton_Radio = tkinter.Radiobutton(fenetre, variable=Ma_Var,
text="Bouton", values=1)

Bouton_Check = tkinter.Checkbutton(fenetre, variable=Ma_Var,
bg="blue", command=fonction)

Bouton_Radio.pack()

Bouton_Check.pack()

Nous avons déjà vu la plupart des options (bg, command, variable, text, values, anchor, etc...) mais nous en avons 3 nouvelles:

- compound : permet d'afficher du texte et une image. On doit indiquer la position de l'image par rapport au texte.
- indicator : modifie l'apparence du boutons. Par exemple, 1 est accompagné d'un indicateur, mais 0 correspond à des boutons poussoirs.
- relief : permet de modifier le relief : FLAT ou RAISED ou SUNKEN ou GROOVE ou RIDGE.

les zones de t茅te utilisatice (Entry)

Saisie = `tkinter.Entry(fenetre, bg="orange", show="*")`

La en cote vous avez une liste tres longue d'option mais seulement deux sont a vous presenter :

- `show` vas remplacer chaque caractere de la ligne de saisie par son parametrie.
- `exportselection`: Mets le t茅te, que vous selectionner, dans le widget, dans votre pince papier. Pour le desactiver, mettre 0.

La barre de menu (Menu)

Il faut penser que l'on peut afficher des menus dans des menu (en cascade)

Creer la Barre de Menu :

`menu_bar = tkinter.Menu(fenetre)`

On peut creer un autre menu avec plusieurs commandes :

`menu_bar`

`Boite_vitesse = tkinter.Menu(menu_bar, tearoff=0)`

`Boite_vitesse.add_command(label="manuel", command=commande)`

`Boite_vitesse.add_command(label="automatique", command=auto)`

L'option `tearoff` permet d'enlever la ligne en pointillier en haut du menu permettant de creer une nouvelle fenetre

On peut ajouter notre menu `Boite_vitesse` a notre `menu_bar`:

`menu_bar.add_cascade(label="Boite", menu=Boite_vitesse)`

On peut également ajouter une commande directement sur notre barre de menu :

menu-bar.add-command (label="Eteindre", command = fenetre.destroy)

Pour finir on ajoute la barre à la fenêtre :
fenetre.config(menu=menu-bar)

Les fenêtres d'erreur (messagebox)

Il existe 6 types de fenêtre mais toute on la même syntaxe :

Avant d'utiliser messagebox, nous avons pris l'habitude de l'importer messagebox :
from messagebox import messagebox.

On commencera par indiquer le type de fenêtre, puis on donnera son titre, son message et enfin des options si besoin.

1) showinfo() sert à afficher des informations.
messagebox.showinfo("Information", "Information for User")
type de fenêtre titre de la fenêtre message.

2) showwarning() sert à afficher des avertissements
messagebox.showwarning("avertissement", "Warning for User")

3) showerror() pour afficher une erreur.
messagebox.showerror("Erreur"; "Erreur 404")

4) askquestion() pour poser une question.

messagebox.askquestion("confirm", "Etes-vous sûre?")

5) askokcancel() pour une confirmation.

messagebox.askokcancel("Redirect", "Redirecting you")

6) askyesno() pour une confirmation.

messagebox.askyesno("Application", "Got it")

Importation de tkinter.

Il existe 3 importation possible :

- from tkinter import * : qui est la version
sociale mais pas celle conseiller.

- import tkinter ; Utiliser dans ce cours mais oblige
à écrire tkinter à chaque fois.

- import tkinter as tk : Recommander, on sera de
quoi on parle sans écrire tkinter (mais juste tk)

Quelques Méthodes

winfo_screenwidth() et winfo_screenheight() permet de
connaître la taille de la fenêtre.

height = fenetre.winfo_screenheight()

minsize() empêche de trop réduire la fenêtre

fenetre.minsize(width=100, height=100)

destroy() permet de détruire la fenêtre.

fenetre.destroy()