ANNEXE : mini-lexique tkinter

Table des matières

I) Fenêtre

- fen=**Tk()** : crée la fenêtre qui s'appellera fen.
- fen.**mainloop()** : provoque le démarrage du gestionnaire d'événements associé à la fenêtre. Cette instruction est nécessaire pour que l'application soit « à l'affût » des clics de souris, des pressions exercées sur les touches du clavier, etc. C'est donc cette instruction qui la met en marche.
- fen.**destroy** : provoque la fermeture de la fenêtre fen.
- fen.**geometry**("400x100") : redimensionne la fenêtre fen en 400 pixels de large et 100 pixels de haut.
- fen.**title**("exemple") : affiche le titre exemple à la fenêtre fen.

II) Principaux Widgets

- **Label** : permet un affichage simple de texte. (éventuellement une image), non modifiable par l'utilisateur, mais modifiable par le programme.
- **Button** : définit un bouton sur lequel on peut cliquer et qui exécute une commande quelconque (une fonction existante dans Python ou une fonction que vous avez codée).

Paramètres de ces widgets :

- ⋆ **text=...** : texte à afficher ;
- ⋆ **fg=...** : couleur du texte ;
- ⋆ **bg=...** : couleur de fond.

Paramètre spécifique au widget Button :

- ⋆ **command** permet de préciser la fonction à lancer lors d'un clic sur le bouton.
  Dans l'exemple précédent, la commande **destroy** actionne la fonction **destroy** qui permet de fermer la fenêtre lorsque le bouton est cliqué.
- ⋆ **eval()** analyse l'expression mathématique entrée par l'utilisateur dans le champ prévu.
  Attention : le paramètre **command** attend un nom de fonction sans argument (donc sans parenthèse).

Paramètres du widget Label :

- ⋆ tex1.**config(...)** : permet de modifier les paramètres du widget tex1, par exemple :
- ⋆ tex1.**config(text="il ne vous reste qu'une vie")**
  permet de modifier l'affichage du **Label** tex1.
- ⋆ tex1.**cget(...)** : renvoie la valeur de l'option demandée sous forme d'une chaine.
- ⋆ tex1.**bind()** : lie une fonction à un widget.

Cette activité est fortement inspirée d'un document élaboré par Christine Agbanrin et Matthieu Gaudré du lycée Alain, Le Vésinet.

- **Checkbutton** : case à cocher qui peut prendre deux états distincts (la case est cochée ou non). Un clic sur ce widget provoque le changement d'état.
  Exemple :

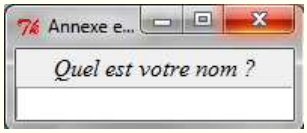
```
from tkinter import *
fen = Tk()
fen.title("Annexe exemples")
# La réponse, stockée dans la variable retour, est de type int
# elle vaudra 0 si la case n'est pas cochée et vaudra 1 sinon
retour=IntVar()
# bou2 : Widget Checkbutton=case à cocher,
bou2=Checkbutton(fen, variable=retour,text="Cochez-moi")
bou2.pack()
fen.mainloop()
# print console
if retour.get()==1 :
print("case cochée")
else :
print("case non cochée")
```

- **Entry** : définit un champ de saisie de texte.
  Il faut donc prévoir une variable permettant de récupérer le texte saisi (ci-dessous la variable s'appelle nom) :
  - ⋆ Nom**=StringVar()** permet de définir une variable qui recevra une chaine de caractère ;
  - ⋆ Nom**=IntVar()** permet de définir une variable qui recevra un entier.

  Le widget **Entry** possède les mêmes paramètres que Nom**Label** sauf **text**, de plus :
  - ⋆ Nom**.get()** permet de récupérer le texte entré par l'utilisateur ;
  - ⋆ Nom**.delete(**i**)** permet d'effacer le contenu du champ de texte à la position i ;
  - ⋆ Nom**.delete(**deb,fin**)** permet d'effacer le contenu du champ de texte entre les indices deb et fin ;
  - ⋆ Nom**.focus()** permet d'obliger le curseur à se placer sur l'**Entry**.

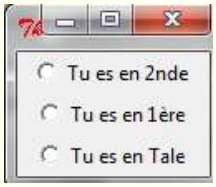
  Exemple :

```
from tkinter import *
fen = Tk()
fen.title("Annexe exemples")
# tex1 : Widget label pour afficher la question
tex1=Label(fen, text="Quel est votre nom ?")
tex1.pack(side=TOP)
tex1.config(font=("times new roman", 12,"italic"))
# La réponse, stockée dans la variable nom, est de type string
nom=StringVar()
# entree : Widget Entry pour lire la réponse
entree=Entry(fen,textvariable=nom,font=("bold",12),width=20)
entree.pack()
fen.mainloop()
print(nom.get()) # le nom entré est imprimé sur la console
```



- **Radiobutton** : définit un ensemble de cases pour une même variable. La variable (nommée ci-dessous etude) retournera les différentes valeurs donnée par **value**.
  Exemple :

```
from tkinter import *
fen = Tk()
fen.title("Annexe exemples")
# La réponse, stockée dans la variable retour, est de type int
etude=IntVar()
# case1, case2, case3 : Widget Radiobutton=cases à cocher
case1=Radiobutton(fen, variable=etude,value=1,text="Tu es en 2nde")
case2=Radiobutton(fen, variable=etude,value=2,text="Tu es en 1ère")
case3=Radiobutton(fen, variable=etude,value=3,text="Tu es en Tale")
case1.pack()
case2.pack()
case3.pack()
fen.mainloop()
print(etude.get()) # la variable etude est affichée à la console
```



Il est possible de donner pour valeur de retour des chaines de caractères. Pour cela, on utilisera etude=**StringVar()**.

- **Canvas** : définit un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner, créer des éditeurs graphiques, pour implémenter des widgets personnalisés (voir le paragraphe suivant pour plus d'informations).
- **Frame** : définit une zone rectangulaire dans la fenêtre, où l'on peut disposer d'autres widgets. Cette zone peut être colorée. Elle peut aussi être décorée d'une bordure.
- **Listbox** : définit une liste de choix proposés à l'utilisateur, généralement présentés dans une sorte de boîte. On peut également configurer la Listbox de telle manière qu'elle se comporte comme une série de « boutons radio » ou de cases à cocher.
- **Menu** : peut être un menu déroulant attaché à la barre de titre, ou bien un menu « pop up » apparaissant n'importe où à la suite d'un clic.
- **Text** : super widget zone de texte (nombreuses possibilités d'interaction).

III) Widget pour dessiner : le widget Canvas

- **Canvas** : définit un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner, créer des éditeurs graphiques, pour implémenter des widgets personnalisés.

  Paramètres :
  - ⋆ **width=...** : épaisseur ;
  - ⋆ **height=...** : hauteur ;
  - ⋆ **bg=...** : arrière plan ;
  - ⋆ **outline=...** : couleur du bord ;
  - ⋆ **fill=...** : définition d'une couleur ;
  - ⋆ **anchor=...** : pour placer (ancrer) un objet (valeurs possibles : **CENTER**, **N**, **S**, **CENTER**, **NW**, **SE**...).

Plusieurs tracés sont possibles dans un canevas :

- can.**create_line(**x1,y1,x2,y2,**fill=**'red') : trace une ligne sur le canevas can du point de coordonnées (x1,y1) au point de coordonnées (x2,y2).
  - ⋆ **fill=...** : pour préciser la couleur choisie ;
  - ⋆ **width=...** : pour préciser la largeur de ligne.

  Tous les paramètres sont facultatifs, ils doivent être séparés par des virgules. Par défaut, le tracé fait en noir sans remplissage.

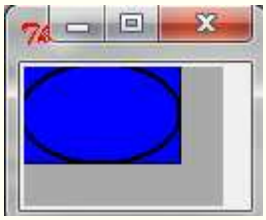
  Pour repérer les coordonnées des points sur l'écran :
  - ⋆ le point en haut à gauche a pour coordonnées (0,0) ;
  - ⋆ le point en bas à droite par exemple dans une fenêtre de 400 ∗ 400 sera le point de coordonnées (400,400) ;
  - ⋆ le point central a pour coordonnées : (largeur de fenêtre /2, hauteur de fenêtre / 2).
- can.**create_text(**x, y, **text=**"mon texte ici", **font=**"Arial 12"**)**
  - ⋆ **fill=...** : couleur de l'écriture ;
  - ⋆ x,y : coordonnées du centre du texte à écrire.
- can.**create_rectangle(**x1,y1,x2,y2**)**

  - ⋆ **outline=...** : couleur du tour du tracé entre guillemets ;
  - ⋆ **fill=...** : couleur de remplissage entre guillemets ;
  - ⋆ x1,y1 : coordonnées du point en haut à gauche ;
  - ⋆ x2,y2 : coordonnées du point en bas à droite.
- can.**create_oval(**x1,y1,x2,y2,**fill=**'couleur', **outline=**'couleur'**)**
  - ⋆ x1,y1,x2,y2 : coordonnées du rectangle circonscrit (qui contient l'ellipse),
    Exemple pour tracer un cercle :
    can**can.create_oval(**x-rayon,y-rayon,x+rayon,y+rayon,**fill=**'couleur', **outline=**'couleur'**)**.
- can.**create_polygon(**x1,y1,x2,y2,x3,y3**)** : c'est comme des tracés de lignes, le dernier côté joint le dernier et le premier points,
  - ⋆ **smooth=True** pour arrondir les angles.

☞De la même façon, essayer **create_arc**...

Exemple

```
from tkinter import *
fen = Tk()
fen.title("Annexe exemples")
can1 = Canvas(fen,bg='dark grey',height=70,width=100)
can1.pack(side=LEFT)
can1.create_rectangle(0,0,80,50, fill="blue")
can1.create_oval(0,0,80,50,outline="black",width=2)
fen.mainloop()
```



IV) Gestion de l'espace dans la fenêtre

- La méthode **pack()** : la fenêtre s'ajuste automatiquement au contenu si on ne précise pas ses dimensions.

  Options de la méthode pack :
  - ⋆ l'option **side** peut accepter les valeurs **TOP**, **BOTTOM**, **LEFT** ou **RIGHT**, pour « pousser » le widget du côté correspondant dans la fenêtre.
  - ⋆ les options **padx** et **pady** permettent de réserver un petit espace autour du widget (exprimé en nombre de pixels).

    padx : espace à gauche et à droite du widget ;

    pady : espace au-dessus et au-dessous du widget.
- La méthode **grid()** : la fenêtre est découpée en un quadrillage virtuel.
  Par exemple toto.**grid(row=**0,**column=**2**)** place le widget toto à la ligne 0 et la colonne 2 (origine en haut à gauche).
  Pour en savoir plus, voir l'ouvrage de G. Swinnen page 95 et suivantes.
- La méthode **place()** : on place un repère sur la fenêtre (origine en haut à gauche). Attention à la portabilité !

ATTENTION : ces méthodes sont mutuellement exclusives.

V) Sitographie :

- Divers widgets prêts à l'emploi :
  http://www.fil.univ-lille1.fr/˜marvie/python/chapitre6.html
- Objets graphiques **Canvas** :
  http://physique.umontreal.ca/˜mousseau/phy1234/notes/notes_52.html