Corrigé de l'exercice du mot de passe

Placement des widgets

La méthode pack()

- la méthode pack()
  - exemple : mon_label.pack()
  - options :
    - side = TOP ou BOTTOM ou LEFT ou RIGHT : cadrage du widget
    - padx, pady : place autour du widget
    - fill = X ou Y ou BOTH : étirement du widget par rapport au parent

```
mon_bouton = Button(ma_fenetre, text = "quitter", font = ("Times", 20),\
height = 5, width = 20, command = ma_fenetre.quit)
mon_bouton.pack(side = RIGHT)
```

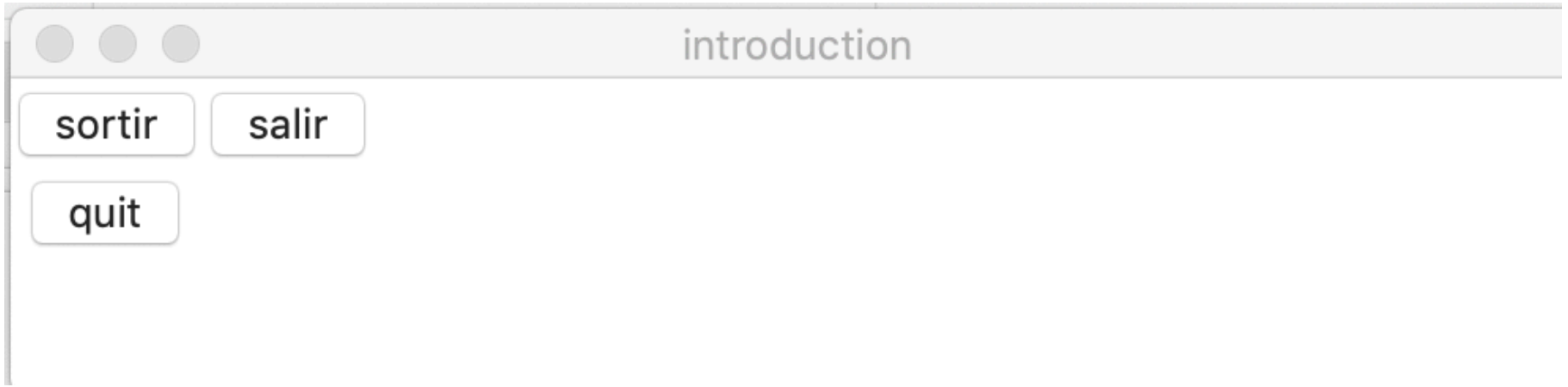
La méthode grid()

- elle découpe la fenêtre ou le cadre en lignes et colonnes :
  - on a donc une **grille** contenant des **cellules**
- paramètres :
  - column, row : numéro de colonne, de ligne (à partir de 0) où l'on veut placer le widget
  - padx, pady, ipadx, ipady : marges extérieures et intérieures
  - columnspan, rowspan : regroupement de cellules
  - sticky : positionnement dans la cellule
    - par défaut centré,
    - dans un coin :"ne" (nord-est), etc.
    - sur un bord : "n" (centré au nord), etc.
    - étiré : "ns" (étiré), etc.

nw	n	ne
w	CENTER	e
sw	s	se

La méthode grid() - Exemple

```
mon_bouton1 = Button(ma_fenetre, text = "sortir", command = ma_fenetre.quit)

mon_bouton2 = Button(ma_fenetre, text = "quit", command = ma_fenetre.quit)

mon_bouton3 = Button(ma_fenetre, text = "salir", command = ma_fenetre.quit)

mon_bouton1.grid(row = 0, column = 0)
mon_bouton2.grid(row = 1, column = 0)
mon_bouton3.grid(row = 0, column = 1)
```



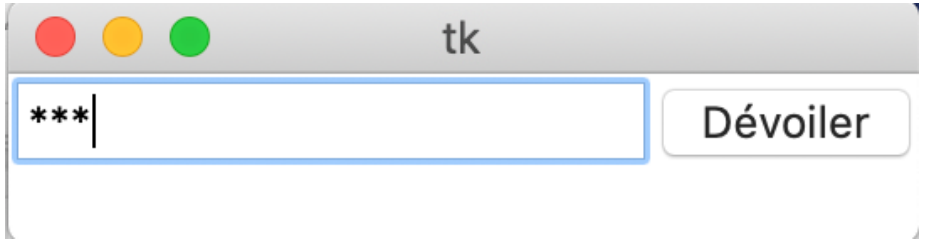
Gestion de la grille

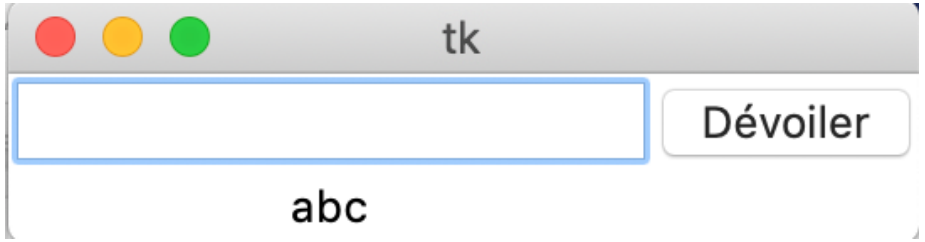
- méthodes associées :
  - grid() : pour faire apparaître un widget
  - grid_remove() : pour faire disparaître un widget (il existe toujours)
  - grid_size() : retourne un tuple avec le nombre de colonnes et de lignes de la grille
- configuration de la taille des lignes et des colonnes
  - avec les méthodes : rowconfigure() et columnconfigure()
  - paramètres :
    - le numéro de la ligne ou de la colonne
    - et :
      - weight : poids relatif d'une ligne ou d'une colonne (pour rendre étirable)
      - pad : marge (en pixel)
      - minsize (en pixel)

exemple
```
f.rowconfigure(0, weight = 2)
f.rowconfigure(1, weight = 1)
```

Exercice

Reprendre l'exercice du mot de passe mais avec le rendu suivant :





Remarque : utilisation d'une variable de contrôle pour gérer l'utilisation d'un objet d'interaction

Retour sur l'utilisation d'un label avec un texte variable

```
from tkinter import *

def compte():
    inter = int(nombre.get())+1
    nombre.set(str(inter))

ma_fenetre = Tk()

nombre = StringVar()
nombre.set(str(0))

mon_label = Label(ma_fenetre, textvariable = nombre)
mon_label.pack()

mon_bouton = Button(ma_fenetre, text = "Ajouter 1", command = compte)
mon_bouton.pack()

ma_fenetre.mainloop()
```

- remarque : la classe StringVar
  - et les méthodes set et get

Retour sur l'utilisation d'un champ (Entry)

```
from tkinter import *

def efface():
    ma_var.set('')

ma_fenetre = Tk()

ma_var = StringVar ()
mon_champ = Entry(ma_fenetre, textvariable = ma_var, width = 50)
mon_champ. pack()

mon_bouton = Button(ma_fenetre, text = "Effacer", command = efface)
mon_bouton.pack()

ma_fenetre.mainloop()
```

Utilisation d'une variable de contrôle pour gérer un objet d'interaction

Dans les exemples précédents :

- pour le label, le programme manipule la variable *nombre* à laquelle est associée le label
  - ==> il peut modifier ce qu'affiche le label par l'intermédiaire de la variable *nombre*.
- pour le champ, le programme manipule la variable *ma_var* associée au champ :
  - ==> il peut récupérer ce que l'utilisateur saisit dans le champ par l'intermédiaire de la variable *ma_var ;*
  - ==> il peut modifier ce qui est affiché dans le champ par l'intermédiaire de la variable *ma_var*.

Utilisation d'une variable de contrôle pour gérer un objet d'interaction

D'autres objets d'interaction utilisent ce système de variable intermédiaire.

Exemple : les cases à cocher


```
from tkinter import *

ma_fenetre = Tk()

coche = IntVar ()
ma_case = Checkbutton(ma_fenetre, text = "recto-verso", width = 20, variable = coche)
ma_case. pack()

ma_fenetre.mainloop()
```

Autre exemple : les boutons

si le texte du bouton doit varier en cours d'exécution, utiliser la propriété *textvariable* plutôt que *text*

Autres classes de la bibliothèque tkinter (2)

Canevas, frame

Les classes déjà vues :

- Tk
- Button
- Label
- Entry

Pendant cette séance :

- Checkbutton
- Radiobutton
- Listbox
- Canvas
- Frame

Widget	Description
Button	Un bouton classique, à utiliser pour provoquer l'exécution d'une commande quelconque.
Canvas	Un espace pour disposer divers éléments graphiques. Ce widget peut être utilisé pour dessiner, créer des éditeurs graphiques, et aussi pour implémenter des widgets personnalisés.
Checkbutton	Une 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.
Entry	Un champ d'entrée, dans lequel l'utilisateur du programme pourra insérer un texte quelconque à partir du clavier.
Frame	Une surface rectangulaire dans la fenêtre, où l'on peut disposer d'autres widgets. Cette surface peut être colorée. Elle peut aussi être décorée d'une bordure.
Label	Un texte (ou libellé) quelconque (éventuellement une image).
Listbox	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	Un menu. Ce 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.
Menubutton	Un bouton-menu, à utiliser pour implémenter des menus déroulants.
Message	Permet d'afficher un texte. Ce widget est une variante du widget Label, qui permet d'adapter automatiquement le texte affiché à une certaine taille ou à un certain rapport largeur/hauteur.
Radiobutton	Représente (par un point noir dans un petit cercle) une des valeurs d'une variable qui peut en posséder plusieurs. Cliquer sur un bouton radio donne la valeur correspondante à la variable, et « vide » tous les autres boutons radio associés à la même variable.
Scale	Vous permet de faire varier de manière très visuelle la valeur d'une variable, en déplaçant un curseur le long d'une règle.
Scrollbar	Ascenseur ou barre de défilement que vous pouvez utiliser en association avec les autres widgets : Canvas, Entry, Listbox, Text.
Text	Affichage de texte formaté. Permet aussi à l'utilisateur d'éditer le texte affiché. Des images peuvent également être insérées.
Toplevel	Une fenêtre affichée séparément, au premier plan.

La classe Checkbutton

❑ quelques propriétés :

- bg, fg : couleur de fond, du texte
- font : police et taille des caractères
- width : taille (en nombre de caractères)
- **variable**
- text, textvariable
- command : command à appeler chaque fois que l'état de la case à cocher change
- state : égal à "normal" ou "disabled" ou "active"

la propriété state existe aussi pour les classes déjà vues

- quelques méthodes :
  - select() et deselect()

exemple : ma_case.select()

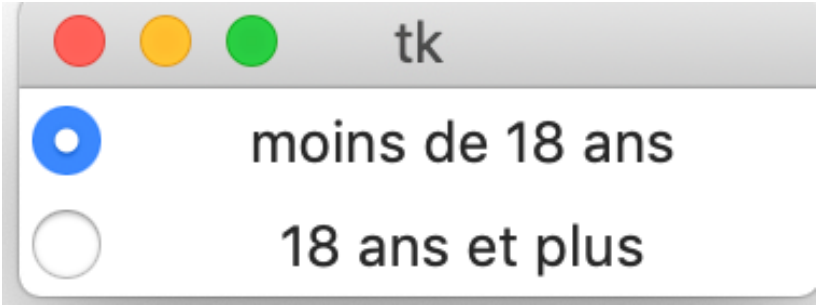
La classe Checkbutton - Exemple

```
from tkinter import *

ma_fenetre = Tk()

coche = IntVar ()
ma_case = Checkbutton(ma_fenetre, text = "recto-verso", width = 20, variable = coche)
ma_case. pack()

ma_fenetre.mainloop()
```

On peut dans le programme :

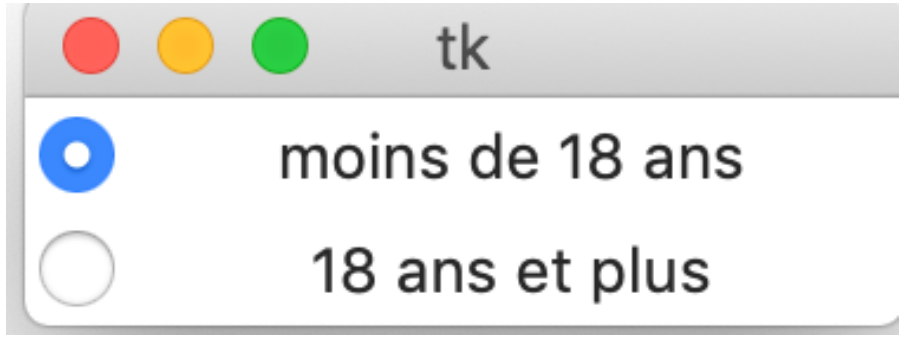
- récupérer la "valeur" de la case à cocher : coche.get()
  - 1 si cochée
  - 0 sinon
- changer le texte de la case à cocher en utilisant la propriété textvariable et une variable de contrôle de type StringVar (comme pour un label)
- associer une procédure à la case à cocher avec la propriété *command*

La classe Radiobutton



- quelques propriétés :
  - bg, fg, font, width
  - **variable**
  - **value**
  - text, textvariable
  - command : command à appeler chaque fois que l'état de la case à cocher change
  - state
- quelques méthodes :
  - select() et deselect()

La classe Radiobutton - Exemple




```
from tkinter import *

def affiche():
    print(categorie.get())

ma_fenetre = Tk()

categorie = StringVar()
categorie.set("mineur")

choix_mineur = Radiobutton(ma_fenetre, text = "moins de 18 ans", \
width = 20, variable = categorie, value = "mineur", command = affiche)

choix_majeur = Radiobutton(ma_fenetre, text = "18 ans et plus", \
width = 20, variable = categorie, value = "majeur", command = affiche)

choix_mineur. pack()
choix_majeur. pack()

ma_fenetre.mainloop()
```

Le regroupement des boutons radio se fait via la variable (ici categorie)

La classe Radiobutton - Exercice

Ecrire un programme qui réalise le système suivant :

La classe Listbox

- quelques propriétés :
  - bg, fg, font, width
  - state
  - height : nombre de lignes affichées (par défaut 10)
  - listvariable : nom de la variable de contrôle du contenu
  - selectmode : 'single' ou 'multiple'
- quelques méthodes :
  - curselection() : renvoie le(s) numéro(s) de la (des) ligne(s) sélectionnée(s) sous la forme d'un tuple
  - insert(…)
  - voir : http://tkinter.fdex.eu/doc/lbw.html

si listvariable = mes_val

- mes_val.set('a' 'b' 'c')
- mes_val.get() renvoie ('a', 'b', 'c')

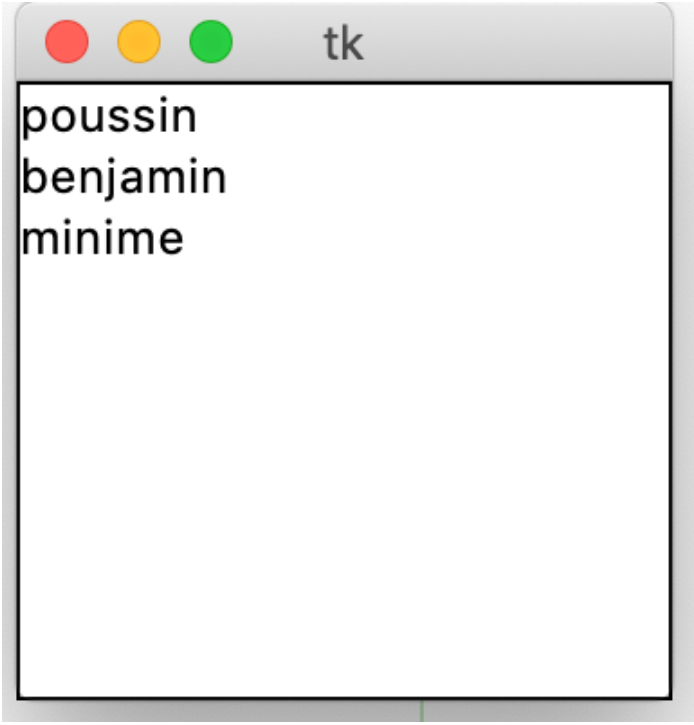
exemple : (0,)

exemple :

l.insert(END, "cadet")

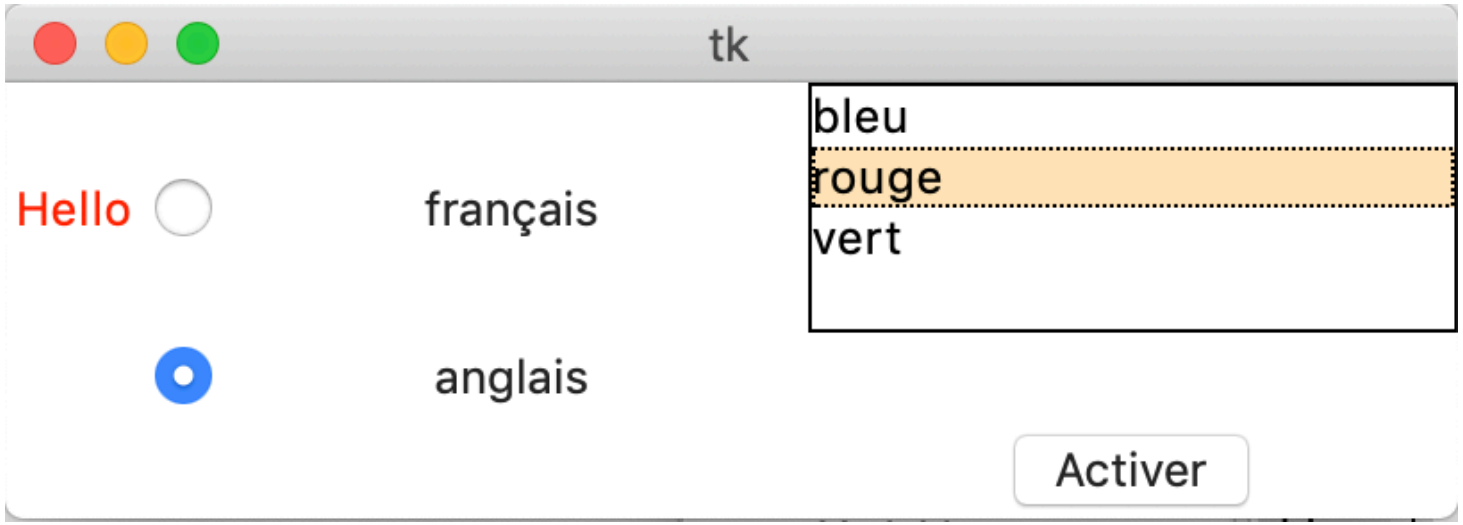
La classe Listbox - Exemple

```
from tkinter import *

ma_fenetre = Tk()

ma_liste = Listbox(ma_fenetre)
ma_liste.pack()

ma_liste.insert(END, "poussin")
ma_liste.insert(END, "benjamin")
ma_liste.insert(END, "minime")

ma_fenetre.mainloop()
```



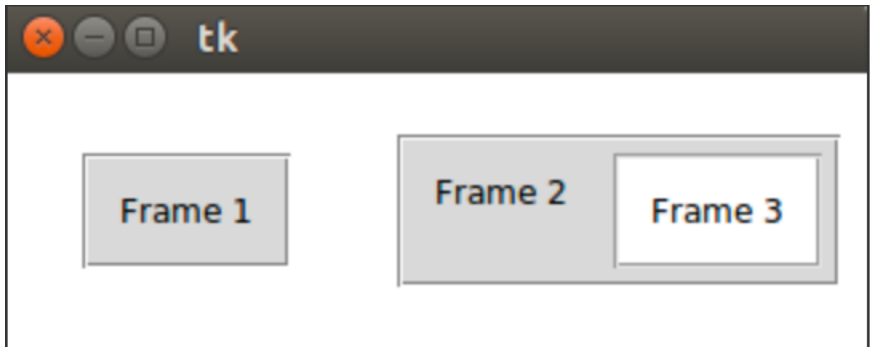
La classe Listbox - Exercice

Compléter le programme précédent :



La classe Frame

- ce sont des conteneurs rectangulaires qui permettent d'organiser des objets d'interaction
- quelques propriétés :
  - bg
  - bd : largeur du cadre
  - height, width

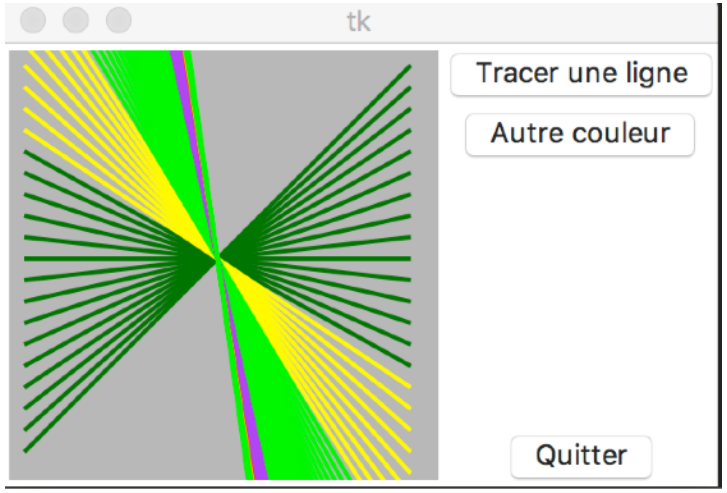


La classe Frame - Exemple

```
from tkinter import *

ma_fenetre = Tk()

mon_cadre1 = Frame(ma_fenetre, width = 50, height = 40, bd = 2, bg = "green")
mon_cadre1.pack()

coche = IntVar()
ma_case = Checkbutton(mon_cadre1, text = "recto-verso", width = 20, variable = coche)
ma_case. pack(side = "top", fill = X)

mon_cadre2 = Frame(ma_fenetre, width = 150, height = 20, bd = 6, bg = "blue")
mon_cadre2.pack()

mon_bouton = Button(mon_cadre2, text = "quitter", font = ("Times", 20),\
height = 5, width = 20, command = ma_fenetre.quit)
mon_bouton.pack()

ma_fenetre.mainloop()
```

La classe Canvas

- surface rectangulaire pour mettre des graphiques ou des images



- quelques propriétés :
  - bg
  - bd : largeur de la bordure
  - height, width

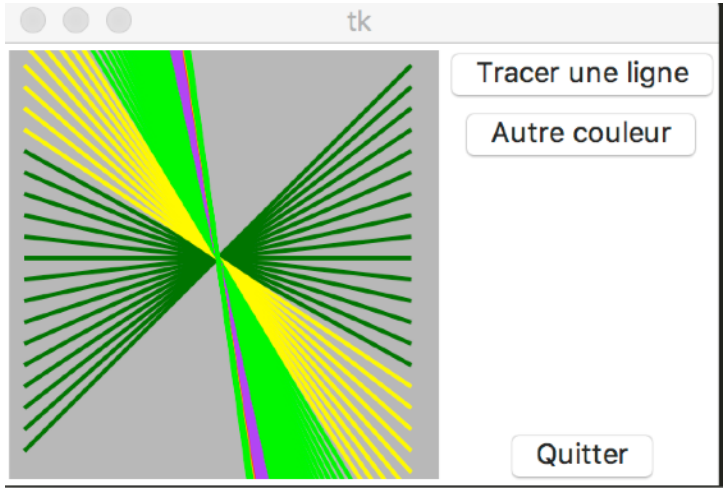
exemple de création d'un objet canvas :

```
can1 = Canvas(ma_fenetre, bg = 'blue', width = 200, height = 200)

can1.pack(side = LEFT)
```

La classe Canvas

quelques méthodes :

- create_line(x0, y0, x1, y1, …, propriétés…) : ligne
- create_rectangle(x0, y0, x1, y1, propriétés…) : rectangle
- create_arc() : arc de cercle
- create_oval() : ovale
- create_polygon() : polygone



quelques propriétés pour les graphiques :

- fill : couleur
- state : "normal", "disabled", "active", "hidden"
- width : épaisseur du trait (en pixel)
- arrow (pour line) : "first", "last", "both"

exemple :

```
can1.create_line(10 190, 190,
10, width = 2, fill = coul)
```

Plus de détails : http://tkinter.fdex.eu/doc/caw.html

La classe Canvas

La méthode create_image(x, y, propriétés…)

- x, y : position de l'image
- propriétés :
  - image, activeimage, disabledimage : images à afficher
  - state : "normal", "disabled", "hidden"

Attention : la bibliothèque tkinter standard n'accepte qu'un petit nombre de formats pour cette image. Choisissez de préférence le format GIF

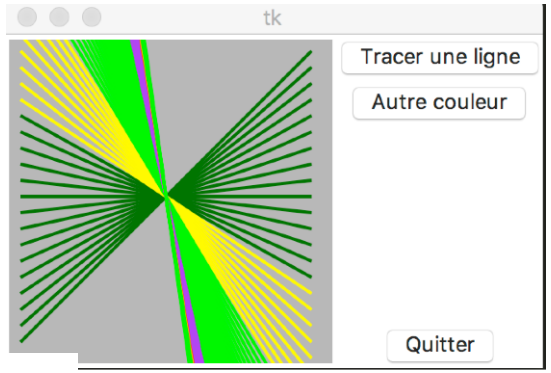
Pour les images gif, il faut utiliser le constructeur PhotoImage

Plus de détails : http://tkinter.fdex.eu/doc/caw.html

```
can1 = Canvas(fen1, width =160, height =160, bg ='white')
photo = PhotoImage(file ='martin_p.gif')
item = can1.create_image(80, 80, image =photo)
Can1.pack()
```

La classe Canevas : exercice

Ecrire un programme qui réalise le système suivant :



On donne les informations suivantes :

```
def dessin_ligne():
    global x0, y0, x1, y1
    can1.create_line(x0, y0, x1, y1, width = 2, fill = coul)
    y0 = y0 - 10
    y1 = y1 + 10

def change_couleur():
    global coul
    palette = ('purple', 'cyan', 'maroon', 'green', 'red', 'blue', 'orange', 'yellow')
    alea = randrange(8)
    print("couleur : ", alea)
    coul = palette[alea]
```

randrange est une fonction de la bibliothèque random

x0, y0, x1, y1 doivent être initialisés dans le programme principal, par exemple à 10, 190, 190, 10

coul doit également être initialisé dans le programme principal

Documentation

http://tkinter.fdex.eu/doc/

documentation officielle de Python (en anglais)
https://docs.python.org/3/library/tkinter.html