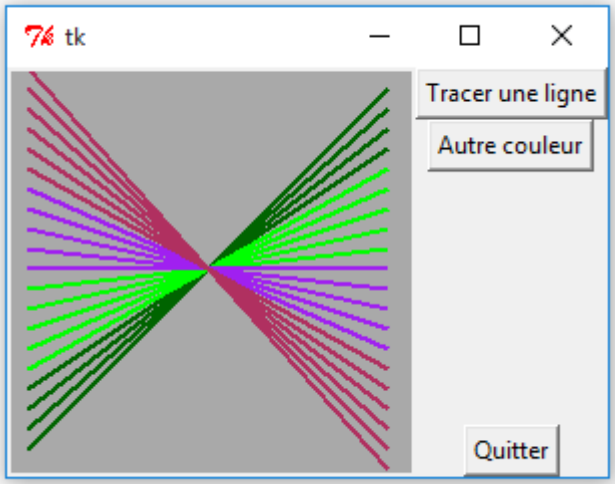
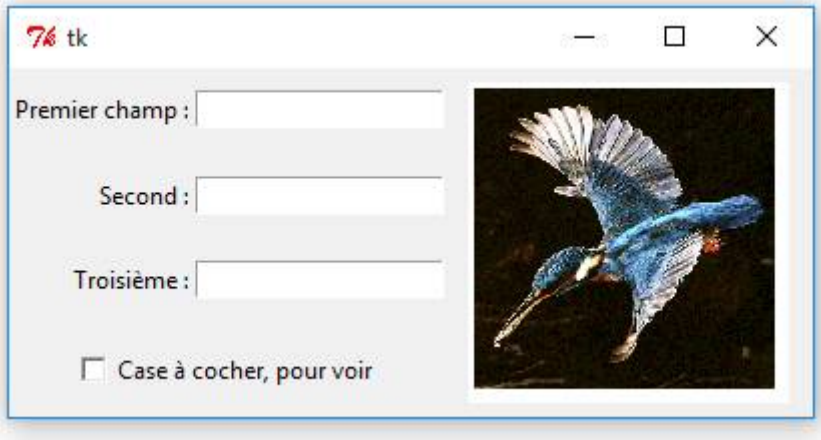
Interface graphique :Tkinter

- ✓ Une interface graphique permet l'interaction entre un utilisateur et un programme.

- ✓ Elle est constituée d'une fenêtre dans laquelle on place des objets que l'on appel des widgets.

- ✓ En Python, le module qui permet de créer une interface graphique se nomme Tkinter. Ce module est installé par défaut lors de l'installation de Python.





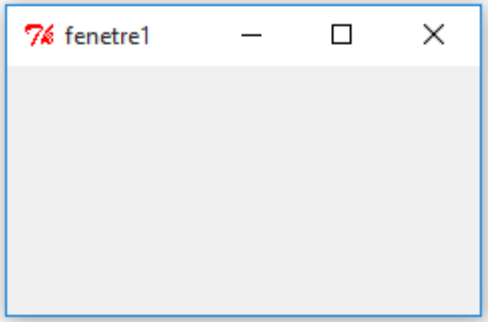
Une partie de ce cours est inspiré du livre de G.Swinnen

Création d'une fenêtre

✓ Enregistrer le script ci-dessous dans un fichier ***fenetre.py*** :

```
from tkinter import*
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
fen.mainloop()
```

Ligne 1: importation du module
Ligne 2 : création de l'objet tkinter nommé fen
Ligne 3 : titre de la fenêtre
Ligne 4 : dimensions de la fenêtre
Ligne 5 : scrutateur d'événements. A ne pas oublier!



Création et placement d'un label dans la fenêtre:

- ✓ Un label est un espace dans la fenêtre prévu pour écrire du texte.
- ✓ Enregistrer le script ci-dessous dans un fichier ***label.py*** :

```
#label + utilisation de la methode pack()

from tkinter import*
#fenetre
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#placement gauche
lab=Label(fen,text='c\'est mon premier texte !',bg='red')
lab.pack(side=TOP) # coller l'objet lab dans la fenêtre.
fen.mainloop()
```

- ✓ Modifier le programme pour obtenir la représentation suivante:

Aide:
Side=BOTTOM
Side=TOP
Side=RIGHT
Side=LEFT

Création et placement d'un canevas dans la fenêtre:

- ✓ Un canevas (canvas) est un espace dans lequel on peut dessiner ou écrire.

- ✓ Enregistrer le script ci-dessous dans un fichier ***canvas.py*** :

```
from tkinter import*
#
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#creer un label
lab=Label(fen,text='texte dans la fenêtre !',bg='red')
lab.pack()
#creer un canvas
can=Canvas(fen,width=500,height=400,bg='light blue')
can.pack()
#un 2eme label
lab2=Label(fen,text='Au dessus mon canvas',fg='red')
lab2.pack()
#
fen.mainloop()
```

Création et placement d'un bouton dans la fenêtre:

✓ Enregistrer le script ci-dessous dans un fichier ***bouton.py*** :

```
from tkinter import*

def ferme(): # fonction appelée lors du clic sur le bouton
    fen.destroy()
#
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#creer un label
lab=Label(fen,text='Mon premièr bouton !',bg='red')
lab.pack()
#creer un canvas
can=Canvas(fen,width=400,height=300,bg='cyan')
can.pack()
#creer un bouton et lui associer une def
b=Button(fen,text='Quitter',command=ferme)
b.pack()
#
fen.mainloop()
```

✓ Lorsqu'on clic sur le bouton, on appel la fonction qui permet de détruire la fenêtre.

Tracer des lignes dans un canvas:

✓ Enregistrer le script ci-dessous dans un fichier ***ligne.py*** :

```
from tkinter import*

def redessine():
    can.delete(ALL)
    lab.config(text='bonjour')
    can.config(bg='light green')
    can.create_line(10,200,400,200,fill='light yellow',arrow=LAST)
#
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#creer un label
lab=Label(fen,text='Les lignes',bg='red')
lab.pack()
#creer un canvas
can=Canvas(fen,width=500,height=400,bg='yellow')
can.pack()
#creer un bouton et lui associer une def
b=Button(fen,text='Afficher',command=redessine)
b.pack()
#creer une ligne dans un canvas
can.create_line(100,150,400,150,fill='blue')
#
fen.mainloop()
```

✓ Que fait ce programme avant le clic sur le bouton puis après le clic?

TD lignes:

Vous allez devoir modifier le programme ci-dessous. Après chaque modification, enregistrez le avec un nom différent pour ne pas l'écraser (*save as).*

Le fichier est disponible sur le disque échange!

```
from tkinter import *
from random import randrange

# définition des fonctions  :
def drawline():
    "Tracé d'une ligne dans le canevas can1"
    global x1, y1, x2, y2, coul
    can1.create_line(x1,y1,x2,y2,width=2,fill=coul)

    # modification des coordonnées pour préparer la ligne suivante :
     y1 = y1-10
     y2 = y2+10

def changecolor():
    "Changement aléatoire de la couleur du tracé"
    global coul
    pal=['purple','cyan','maroon','green','red','blue','orange','yellow']
    c = randrange(8)        # => génère un nombre aléatoire de 0 à 7
    coul = pal[c]

#------ Programme principal -------
# les variables suivantes seront utilisées de manière globale :
x1, y1, x2, y2 = 10, 190, 190, 10   # coordonnées de la ligne
coul = 'dark green'                 # couleur de la ligne

# Création fenêtre principale :
fen1 = Tk()
# création des widgets  :
can1 = Canvas(fen1,bg='dark grey',height=200,width=200)
can1.pack(side=LEFT)
#
bou1 = Button(fen1,text='Quitter',command=fen1.destroy)
bou1.pack(side=BOTTOM)
#
bou2 = Button(fen1,text='Tracer une ligne',command=drawline)
bou2.pack()
#
bou3 = Button(fen1,text='Autre couleur',command=changecolor)
bou3.pack()
#
fen1.mainloop()
```

Travail a effectuer: diapo suivante!

s.loutfi

TD lignes:

1. Modifier le programme pour ne plus avoir que des lignes de couleur cyan, maroon et green.

2. Modifier le programme pour que toutes les lignes tracées soient horizontales et parallèles.

3. Agrandir le canevas de manière à lui donner une largeur de 500 unités et une hauteur de 650. Modifier également la taille des lignes, afin que leurs extrémités se confondent avec les bords du canevas.

4. Ajouter une fonction drawline2( ) qui tracera deux ligne rouges en croix au centre du canevas : l'une horizontale et l'autre verticale.

Ajouter également un bouton portant l'indication « viseur ». Un clic sur ce bouton devra provoquer l'affichage de la croix.

s.loutfi

Tracer des cercles et des rectangles dans un canevas:

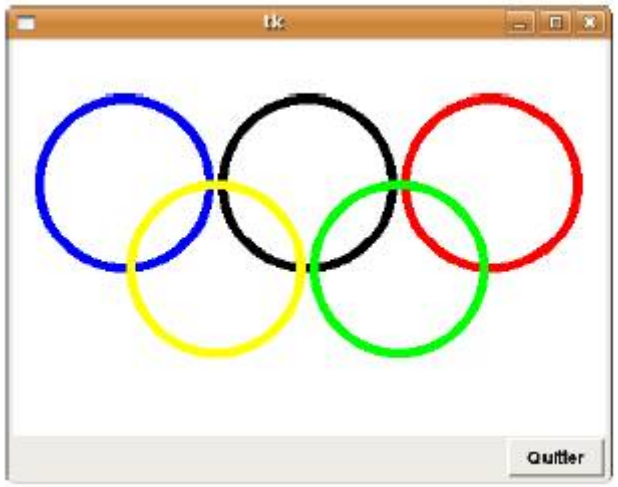
✓ Enregistrer le script ci-dessous dans un fichier ***cercle.py*** :

```
from tkinter import*

def texte():
    can.delete(ALL)
    #creer texte:
    txt=can.create_text(300,250,text='Texte dans le canvas!',font='arial 20')
#
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#creer un label
lab=Label(fen,text='Texte dans la fenêtre !',bg='red')
lab.pack()
#creer un canvas
can=Canvas(fen,width=500,height=400,bg='light blue')
can.pack()
#creer un bouton et lui associer une def
b=Button(fen,text='Afficher',command=texte)
b.pack()
#creer un cercle dans un canvas
can.create_oval(100,190,120,210,fill='red')
#creer un rectangle dans un canvas
can.create_rectangle(10,10,150,150, fill='yellow')
#
fen.mainloop()
```

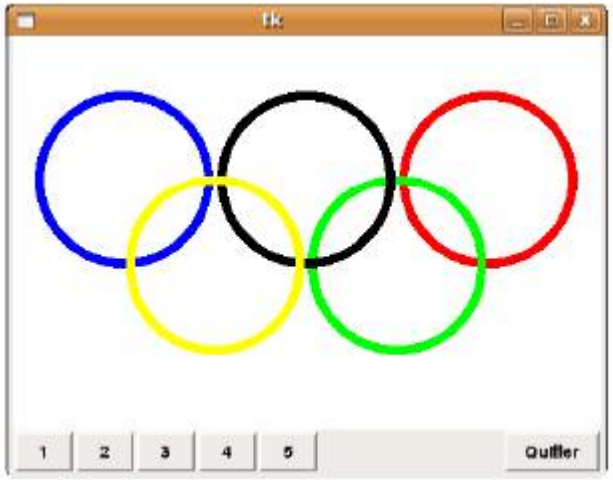
✓ Que fait ce programme avant le clic sur le bouton puis après le clic?

TD cercles:

Créer un programme qui dessinera les cinq anneaux olympiques dans un rectangle de fond blanc (white).Utiliser l'argument outline à la place de fill pour la couleur des anneaux.
Un bouton « quitter » doit permettre de fermer la fenêtre.



Modifier le programme précédent en y ajoutant cinq boutons. Chacun de ces boutons provoquera le tracé d'un anneau.

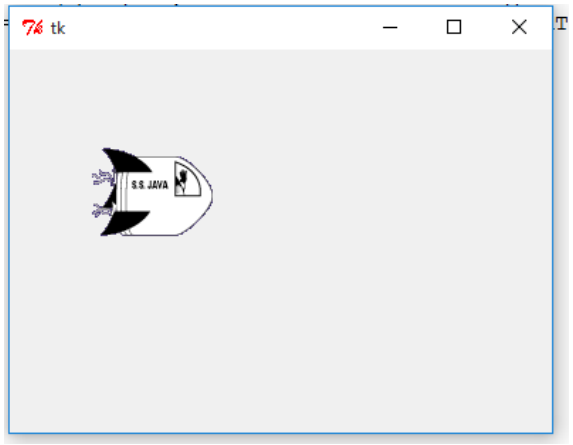


s.loutfi

Insérer une image:

✓ Enregistrer le script ci-dessous dans un fichier ***photo.py*** :

```
from tkinter import *

fen=Tk()
#
can=Canvas(fen)
can.pack()
#creer un objet PhotoImage
photo=PhotoImage(file='fusee.gif')

#coordonnées du centre de l'image(100,100)
can.create_image(100,100,image=photo) #
#
fen.mainloop()
```

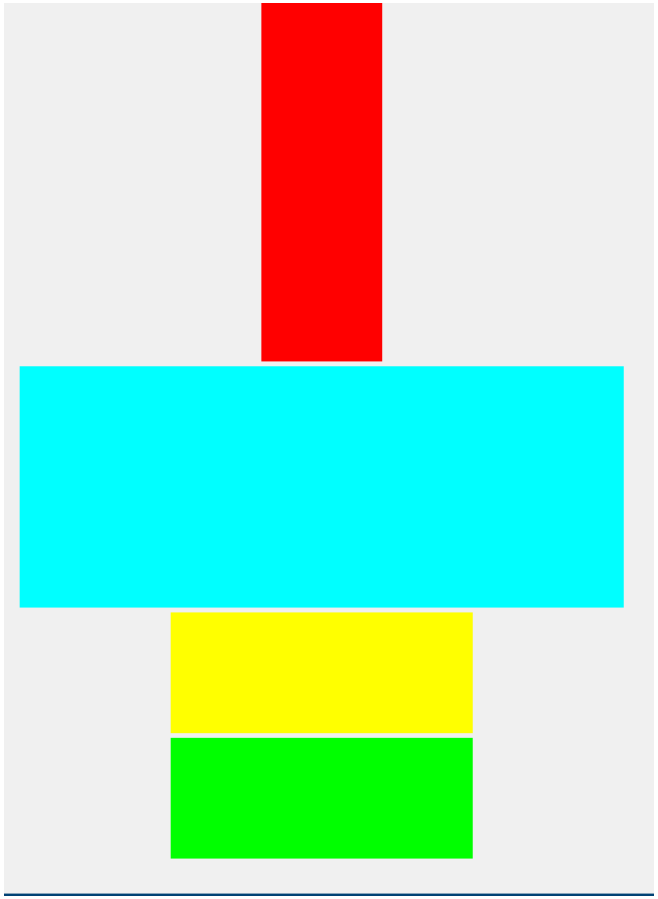


✓ Attention: les formats d'images acceptées sont: .gif ; .png ; pgm et ppm

Méthode de placement: pack()

- ✓ Enregistrer le script ci-dessous dans un fichier ***placer.py*** :
- ✓ **Pour la méthode pack, décommenter uniquement les lignes can.pack()**

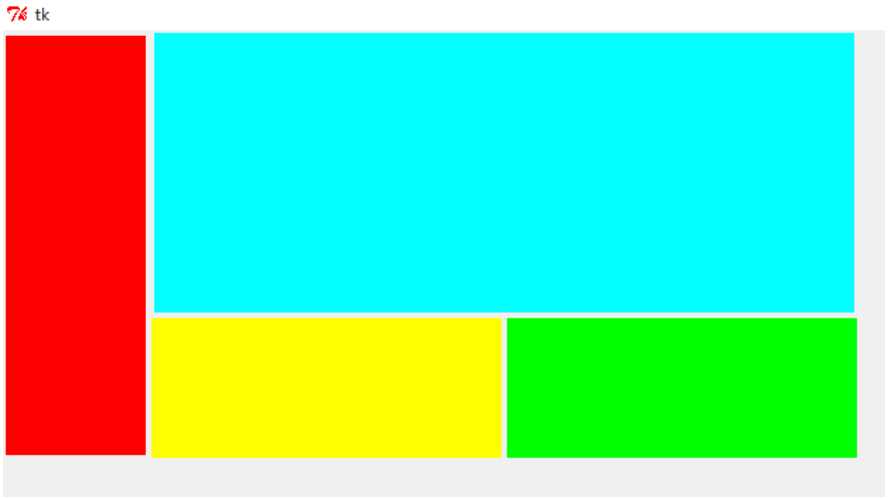
```
from tkinter import*

fen=Tk()
fen.geometry('600x500')
#
can1=Canvas(fen,width=100,height=300,bg='red')

#can1.pack()
#can1.grid(row=0,column=0,rowspan=2)
#can1.place(x=100,y=50)
#
can2=Canvas(fen,width=500,height=200,bg='cyan')
#can2.pack()
#can2.grid(row=0,column=1,columnspan=2)
#can2.place(x=220,y=50)
#
can3=Canvas(fen,width=250,height=100,bg='yellow')
#can3.pack()
#can3.grid(row=1,column=1)
#can3.place(x=220,y=250)
#
can4=Canvas(fen,width=250,height=100,bg='green')
#can4.pack()
#can4.grid(row=1,column=2)
#can4.place(x=470,y=250)
#
fen.mainloop()
```



Méthode de placement: grid()

- ✓ Modifier le fichier ***placer.py*** comme indiqué ci dessous:
- ✓ **Pour la méthode pack, décommenter uniquement les lignes can.grid(…….)**

```
from tkinter import*

fen=Tk()
fen.geometry('600x500')
#
can1=Canvas(fen,width=100,height=300,bg='red')

#can1.pack()
#can1.grid(row=0,column=0,rowspan=2)
#can1.place(x=100,y=50)
#
can2=Canvas(fen,width=500,height=200,bg='cyan')
#can2.pack()
#can2.grid(row=0,column=1,columnspan=2)
#can2.place(x=220,y=50)
#
can3=Canvas(fen,width=250,height=100,bg='yellow')
#can3.pack()
#can3.grid(row=1,column=1)
#can3.place(x=220,y=250)
#
can4=Canvas(fen,width=250,height=100,bg='green')
#can4.pack()
#can4.grid(row=1,column=2)
#can4.place(x=470,y=250)
#
fen.mainloop()
```



Row=0 ou 1 → ligne= 0 ou 1

Column =0 , 1 ou 2 →colonne = 0,1 ou 2

Rowspan =2→ nombre de lignes couvertes =2

Columnspan =2→ nombre de colonnes couvertes=2

Méthode de placement: place()

- ✓ Modifier le fichier ***placer.py*** comme indiqué ci-dessous:
- ✓ **Pour la méthode pack, décommenter uniquement les lignes can.place(x=..,y=..)**

```
from tkinter import*

fen=Tk()
fen.geometry('600x500')
#
can1=Canvas(fen,width=100,height=300,bg='red')

#can1.pack()
#can1.grid(row=0,column=0,rowspan=2)
#can1.place(x=100,y=50)
#
can2=Canvas(fen,width=500,height=200,bg='cyan')
#can2.pack()
#can2.grid(row=0,column=1,columnspan=2)
#can2.place(x=220,y=50)
#
can3=Canvas(fen,width=250,height=100,bg='yellow')
#can3.pack()
#can3.grid(row=1,column=1)
#can3.place(x=220,y=250)
#
can4=Canvas(fen,width=250,height=100,bg='green')
#can4.pack()
#can4.grid(row=1,column=2)
#can4.place(x=470,y=250)
#
fen.mainloop()
```

Les coordonnées x,y sont exprimées en pixels

LES ÉVÉNEMENTS:

Clic de la souris:

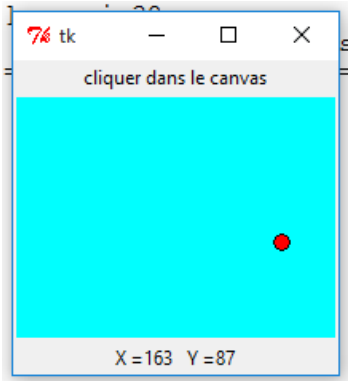
✓ Enregistrer le script ci-dessous dans un fichier ***clic.py*** :

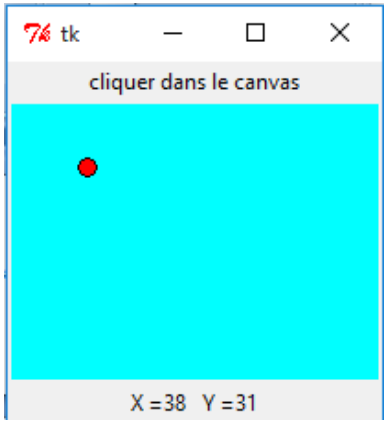
```
from tkinter import *

def pointeur(event):
    X=event.x
    Y=event.y
    lab2.config(text = "X =" + str(X) +"    Y =" + str(Y))
#
fen = Tk()

#label qui donne la consigne
lab1=Label(fen,text='cliquer dans le canvas')
lab1.pack()

#Le canvas
can = Canvas(fen, width =200, height =150, bg="cyan")
can.pack()

#Lier le bouton gauche de la souris au can
#et appeler la fonction pointeur
can.bind("<Button-1>", pointeur)
#
lab2 = Label(fen,text='coordonnées du clic')
lab2.pack()
#
fen.mainloop()
```

Lorsqu'on clique dans le canvas, on passe le paramètre event a la fonction pointeur:

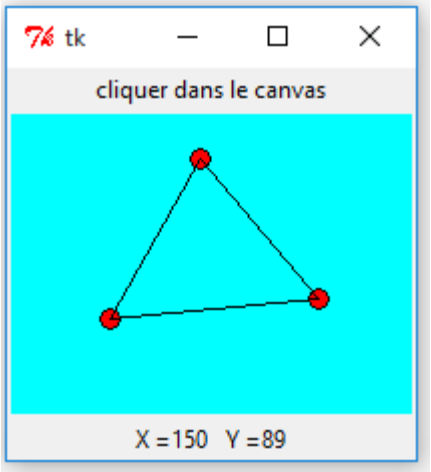
Clic de la souris:

- ✓ Modifier le script ***clic.py*** de tel sorte que le clic de la souris dans le canvas fasse apparaitre un petit cercle rouge là ou on clique:





- ✓ Modifier le script de tel sorte que le programme trace un triangle dont les sommets sont donnés par 3 clics de la souris dans le canvas



Appui sur une touche:

- ✓ Enregistrer le script ci-dessous dans un fichier ***touche.py*** :
- ✓ N.B: dans ce script on utilisera la touche « UP »

```
from tkinter import*

def change(evt):#incrementer le compteur par appuie sur la touche Up
    global x
    if  (evt.keysym)=="Up":
        x=x+1
    lab.config(text=x)

fen=Tk()
#
x=0
#
lab1=Label(fen,text='appuyer sur la touche UP')
lab1.pack()
#
lab=Label(fen,text=x)
lab.bind("<Key>",change)
lab.focus_set()
lab.pack()
#
fen.mainloop()
```

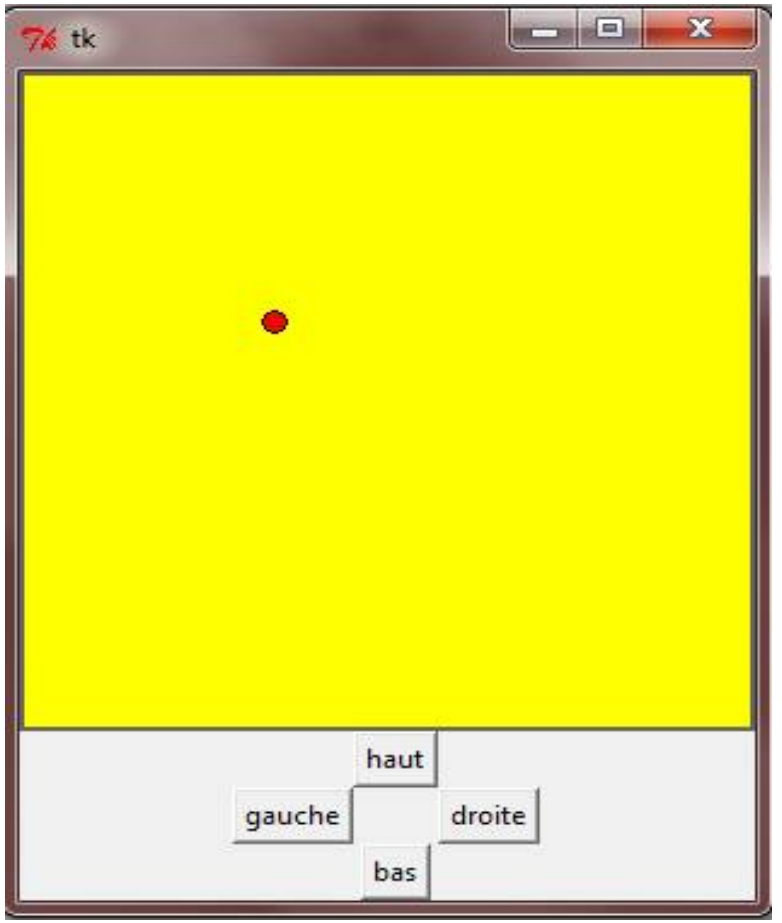
Lien entre l'appui sur une touche et la label. Un événement est envoyé a la fonction change

- ✓ Modifier le programme pour ajouter une fonction qui permet de désincrémenter le compteur par appui sur la touche « DOWN » :

Appui sur une touche:

✓ Créer une fenêtre dans laquelle on peut déplacer un cercle grâce aux boutons ou grâce aux touches du clavier

```
from tkinter import*
x=100
y=100
def droite():
    global x,y
    can.delete(ALL)
    x=x+10
    can.create_oval(x,y,x+10,y+10,fill='red')
def gauche():
    #remplacer pass par votre code
    pass
def bas():
    #remplacer pass par votre code
    pass
def haut():
    #remplacer pass par votre code
    pass
def traite(evt):
    if (evt.keysym)=="Right":
        droite()

fen=Tk()

can=Canvas(fen,width=300,height=300,bg='yellow')
can.grid(row=0,column=1)

fram=Frame(fen)
fram.grid(row=1,column=1,rowspan=2)

b1=Button(fram,text="haut",command=haut)
b2=Button(fram,text="gauche",command=gauche)
b3=Button(fram,text="droite",command=droite)
b4=Button(fram,text="bas",command=bas)

b1.grid(row=0,column=1)
b2.grid(row=1,column=0)
b3.grid(row=1,column=2)
b4.grid(row=2,column=1)

can.bind("<Key>",traite)
can.focus_set()
fen.mainloop()
```



Méthodes *after* et *after_cancel*

after: permet l'appel d'une méthode après un certain temps écoulé en millisecondes

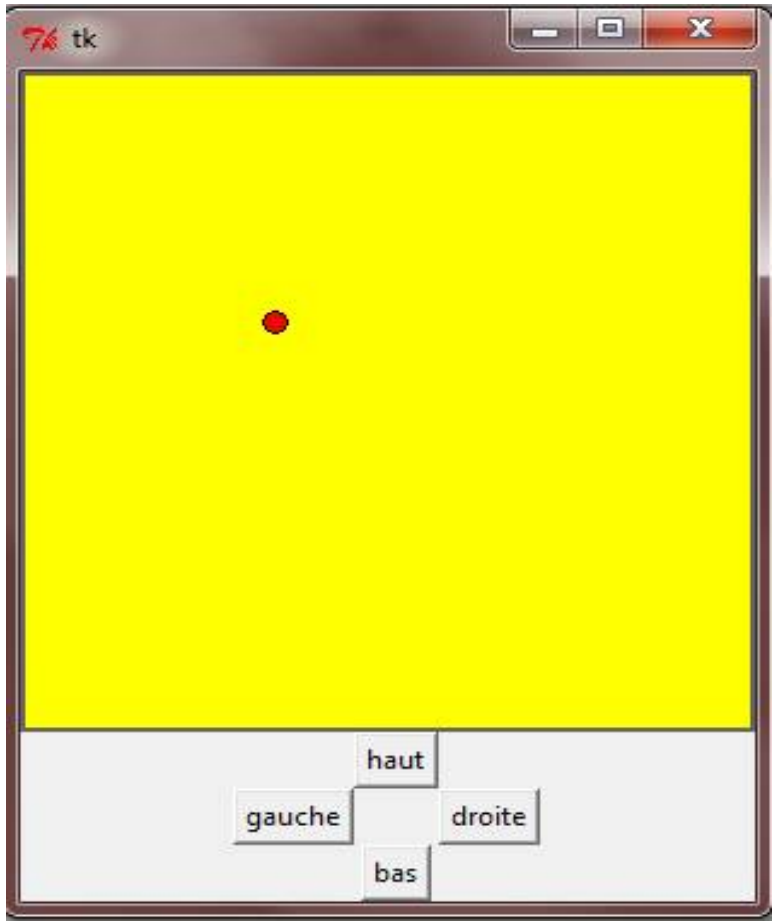
```
from tkinter import*

def apres():
    global x,a
    x=x+1
    lab.config(text=x)
    a=lab.after(1000,apres)

def arret():
    lab.after_cancel(a)

# fenêtre
fen =Tk()
fen.title('After')
#label dans la fenêtre
x=0
lab =Label(fen,text=x)
lab.pack()
#bouton associé à la fonction apres
b=Button(fen,text='Depart',command=apres)
b.pack()
#bouton associé a la fonction arret
b1=Button(fen,text='Arret',command=arret)
b1.pack()
#
fen.mainloop()
```

after_cancel: permet d'arrêter les appels récursifs à la méthode ***after*** en utilisant l'identifiant renvoyé par ***after*** :

Application de la méthode *after*

✓ Modifier le programme précédent afin d'automatiser le déplacement du cercle:

- Un appui sur le bouton « droite » par exemple provoque le déplacement continu du cercle vers la droite,
- Un appui sur un autre bouton provoquera le changement de direction du déplacement.



Le widget Entry:

Un champs de saisie Entry permet à l'utilisateur de rentrer une ligne de texte
Deux méthodes importantes:
entree.get() : lecture du texte entré
entree.insert (position,texte) : insertion d'un texte

```
from tkinter import*

def affiche(evt):
    ent2.delete(0,END)
    var=ent1.get()
    ent2.insert(0,var)

#fenetre
fen=Tk()
fen.title('fenetre1')
fen.geometry('600x500')
#placement champ d'entrée gauche
ent1=Entry(fen)
ent1.pack(side=LEFT)
#placement champ d'entrée droit
ent2=Entry(fen)
ent2.pack(side=RIGHT)
#placement label du haut
lab=Label(fen,text='entrer une valeur dans le champ de gauche ',bg='blue',font=('20'))
lab.pack(side=TOP)
#placement label du bas bas
lab=Label(fen,text='appuyer sur la touche Entrée',bg='yellow',font=('20'))
lab.pack(side=BOTTOM)
#
ent1.bind("<Return>",affiche)
ent1.focus_set()

fen.mainloop()
```

Application du champ Entry:

1. Réaliser l'interface graphique ci-dessous.
2. Écrire un programme qui fasse la conversion des degrés Celsius vers les degrés Fahrenheit en appuyant sur la touche *Return, et* vice-versa.

Tag_bind:

Permet d'attribuer une étiquette à un objet afin de le lier à un événement.

```
from tkinter import*

def afficheR(event):
        lab.config(text='carre_rouge')
def afficheV(event):
        lab.config(text='carre_vert')

#
fen=Tk()
fen.geometry("600x500")
fen.title('TAG_BIND')
#
lab=Label(fen,text='affiche')
lab.pack()
#
can=Canvas(fen,width=400,height=450,bg='cyan')
can.pack()
#
can.create_rectangle(10,10,50,50,fill='red',tags='carre_rouge')
#
can.create_rectangle(100,100,150,150,fill='green',tags='carre_vert')

#
can.tag_bind('carre_rouge','<1>',afficheR)
can.tag_bind('carre_vert','<1>',afficheV)

#
fen.mainloop()
```

Etiquette

Plus d'info sur Tkinter et les autres widgets:

https://wiki.python.org/moin/TkInter

http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html

http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

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