Master chiffre : Python

Règle du jeu : l'ordinateur choisit un nombre entre 1000 et 9999.

L'utilisateur propose un nombre. Si un des chiffres du nombre proposé correspond à un des chiffres du nombre choisi par l'ordinateur (même valeur, même place), le programme affiche ce chiffre. L'utilisateur propose un nouveau nombre, jusqu'à ce que les 4 chiffres composant le nombre de l'ordinateur, soient trouvés.

Première étape

Nous commençons par faire le même programme que celui fait avec Scratch, sans interface graphique, en mode console.

```
import random

#Fonction analyser : compare le  nombre du joueur à celui de l'ordinateur

def analyser(nombre, nombre2):

    #Une liste à afficher après l'analyse : * correspond à une case vide

    chiffres = ["*","*","*","*"]

    #Une boucle sur la longueur du nombre choisi par l'ordinateur soit sur 4 chiffres

    for i in range (len(nombre)):

        #nombre et nombre2 pointent sur une chaine de caractères:

        #on accède à chaque caractère en indiquant son rang.

        #On compare dans les deux chaines, les caractères de même rang

        if nombre[i] == nombre2[i]:

            #Bon chiffre à la bonne place : on modifie la liste chiffres en plaçant

            # le caractère nombre[i] dans la case [i] de la liste.

            chiffres[i] = nombre[i]

            trouve=trouve+1

    #retour au programme appelant en renvoyant la référence à la liste chiffres et

    #au nombre trouve.
```

```
    return chiffres,trouve


#programme principal
jouer = True
#on génère un nombre aléatoire et on traduit  sa valeur numérique en chaine de caractères
nombre_ordi = str((random.randint(1000, 9999))) print("jouons !")
print(nombre_ordi) #pour les essais, à supprimer ensuite
while jouer:
    #input affiche le texte et attend que le joueur tape une réponse.
    #Cette réponse est renvoyée sous forme d'une chaine de caractères.
    nombre_joueur = input("Entrez un nombre entre 1000 et 9999 : ")
    #Ici on permet au joueur de sortir avant la fin du jeu.
     if nombre_joueur == "fin":
         break      #on sort de la boucle while. Le joueur veut arrêter avant la fin
    #Appel à la fonction d'analyse et affichage du résultat
     afficheNombre, tr = analyser(nombre_ordi, nombre_joueur)
    #Si tr = 4, on sort de la boucle de jeu sinon on affiche la liste chiffres
   if tr == 4:
         print("Vous avez gagné la manche et trouvé les 4 chiffres")
         jouer = False  #cela met fin à la boucle while
     else:
         print(afficheNombre)
```

Exemple d'exécution :

```
jouons !
8700

Entrez un nombre entre 1000 et 9999 : 1234
['*', '*', '*', '*']

Entrez un nombre entre 1000 et 9999 : 1230
['*', '*', '*', '0']

Entrez un nombre entre 1000 et 9999 : 1200
['*', '*', '0', '0']

Entrez un nombre entre 1000 et 9999 :
```

Le nombre à trouver est affiché le temps des essais après le Jouons.

Deuxième étape

Maintenant que le moteur de notre jeu est prêt, nous allons créer une interface graphique avec le module tkinter.

Tkinter est un module qui permet de créer une fenêtre :

Import tkinter as tk

fen=tk.Tk()

Dans cette fenêtre on peut ajouter des objets boutons, des objets labels pour permettre l'affichage de textes par l'ordinateur, des objets zones de texte pour permettre à l'utilisateur d'entrer des données et d'autres objets. Tous ces objets s'appellent des **widgets**.

Des **évènements** peuvent être associés aux widgets (clic de souris, appui sur une touche). Lorsque l'évènement arrive, une **fonction (callback)** est exécutée en réponse à l'évènement.

Avec tkinter, nous ne pouvons pas, comme nous l'avons fait avec Scratch, envoyer un « évènement personnalisé : afficher » à un bouton ou un label.

Nous devons donc trouver une autre façon de faire.

Pour notre jeu, nous avons besoin de :

- 4 labels pour afficher les chiffres trouvés ou une * si le chiffre n'a pas été trouvé.
- Un label pour les messages de l'ordinateur : « Cliquer pour commencer une partie », « Donnez un nombre entre1000 et 9999 », « Donnez un nouveau nombre », « Vous avez gagné »
- Une zone de texte dans laquelle le joueur tape le nombre qu'il propose
- Un bouton pour démarrer une partie, lançant le tirage d'un nombre aléatoire entre

1000 et 9999.

Pour aider le programmeur à placer ces objets dans la fenêtre, tkinter fournit des **gestionnaires de positionnement**. L'un d'eux est une grille.

Ce gestionnaire « découpe » chaque fenêtre ou cadre en une grille formée de lignes et de colonnes. **Lire attentivement** la documentation suivante pour comprendre comment fonctionne ce gestionnaire : http://tkinter.fdex.eu/doc/gp.html#w.grid

Création de l'interface

Créons nos 4 labels.

```
import tkinter as tk

#construction de la fenêtre

fen=tk.Tk()

# on indique dans le constructeur du label, l'objet  auquel il est attaché : ici la fenêtre fen

millier = tk.Label(fen)

centaine = tk.Label(fen)

dizaine = tk.Label(fen)

unite = tk.Label(fen)

#placement des labels dans la grille

millier.grid(row = 0, column = 0)

centaine.grid(row = 0, column = 1)

dizaine.grid(row = 0, column = 2)

unite.grid(row = 0, column = 3)

#fin boucle d'existence de la fenêtre : elle se termine lorsqu'on clique sur la croix en haut à droite de la fenêtre

fen.mainloop()
```

Résultat :

Les labels n'apparaissent pas, car comme il n'y a rien d'écrit dedans, tkinter ne les affichent pas.

Voir les paramètres du constructeur de label : http://tkinter.fdex.eu/doc/labw.html#Label

Remplaçons les lignes de construction des labels par :

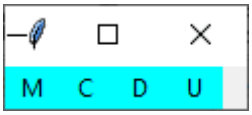
millier = tk.Label(fen, **text = "M", bg="cyan", height=1, width = 3**)

centaine = tk.Label(fen, **text = "C", bg="cyan", height=1, width = 3**)

dizaine = tk.Label(fen, **text = "D", bg="cyan", height=1, width = 3**)

unite = tk.Label(fen, **text = "U", bg="cyan", height=1, width = 3**)

Résultat:



Les labels sont bien visibles mais ils sont collés les uns aux autres et collés au bord de la fenêtre. **Voir les paramètres** padx, pady du gestionnaire grid.

millier.grid(row = 0, column = 0, **padx** = 5, **pady** = 5)

centaine.grid(row = 0, column = 1, **padx = 5, pady = 5**)

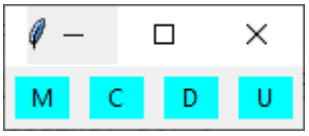
dizaine.grid(row = 0, column = 2, **padx = 5, pady = 5**)

unite.grid(row = 0, column = 3, **padx = 5, pady = 5**)

Les labels sont séparés les uns des autres de 5 pixels, au-dessus, en dessous et sur les côtés.

Ils sont centrés dans leur cellule.

Résultat:



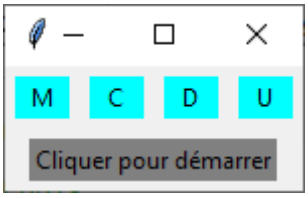
Ajoutons maintenant un label pour les messages de l'ordinateur. (Au-dessus de l'instruction fen.mainloop)

LabelOrdi = tk.Label(fen, text="Cliquer pour démarrer" , bg ="grey")

LabelOrdi.grid(row=1,column=0, **columnspan**=4, padx = 5, pady = 5)

Ce label s'étend sur les 4 colonnes de la grille. Il est centré sur ces 4 colonnes.

Résultat:



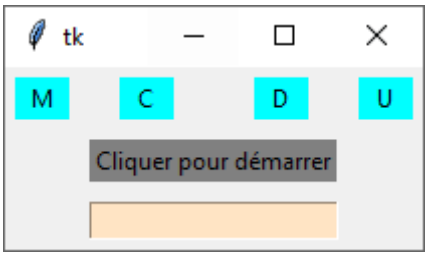
Le LabelOrdi s'étend sur les quatre colonnes, il est centré sur cet ensemble et sa taille est ajustée à son contenu.

Ajoutons une zone d'entrée, pour que le joueur puisse soumettre son nombre.

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

```
ChampJoueur = tk.Entry(fen,  bg ="bisque", fg="maroon")
ChampJoueur.grid(row=2,column=1,columnspan=2, padx = 5, pady = 5)
```

Résultat :



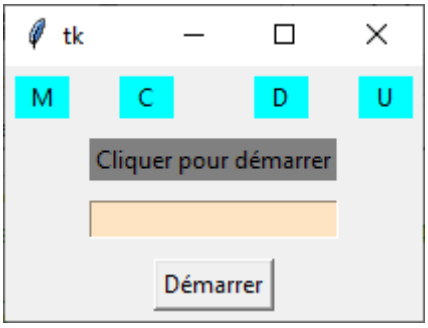
On voit ici que l'espace entre C et D a été agrandi pour que la zone de texte puisse se caler sur la largeur de la deuxième colonne. On essaiera d'arranger ça plus tard.

Ajoutons enfin le bouton permettant de lancer le jeu.

http://tkinter.fdex.eu/doc/bw.html#Button

```
demarrer=tk.Button(fen,text="Démarrer")
demarrer.grid(row=3,column=1, columnspan=2, padx = 5, pady = 5)
```

Résultat :



Les évènements auxquels l'application doit répondre

- Un clic sur « Démarrer » doit générer un nombre entre 1000 et 9999, modifier le texte afficher dans « labelOrdi »
- Lorsque le joueur a tapé son nombre, un appui sur la touche <Return> doit lancer l'analyse du nombre proposé et afficher une * ou le chiffre trouvé dans un des labels « chiffres »

Le bouton « Démarrer »

Nous ajoutons dans son constructeur un paramètre **command=nom fonction**.

Cette fonction sera exécutée lors d'un clic sur le bouton.

Dans cette fonction nous devons générer un nombre. Le problème est que nous ne pouvons pas récupérer ce nombre par un return de la fonction. Une fonction callback peut renvoyer une valeur, mais nous ne pouvons écrire : nombre = maFonction()

Nous devons donc créer un **nombre global**, que la fonction viendra directement remplir par le nombre aléatoire.

Autre problème : la modification du texte dans le LabelOrdi. Nous ne pouvons pas le faire directement par une instruction de la forme : LabelOrdi.text = 'blablabla'

Nous devons lier le texte affiché dans le label à une variable **Stringvar**

http://tkinter.fdex.eu/doc/ctrvar.html?highlight=stringvar

Modification de la création du LabelOrdi et du bouton Démarrer :

#----------

TextOrdi = tk.StringVar()

TextOrdi.set("Cliquez sur démarrer pour jouer")

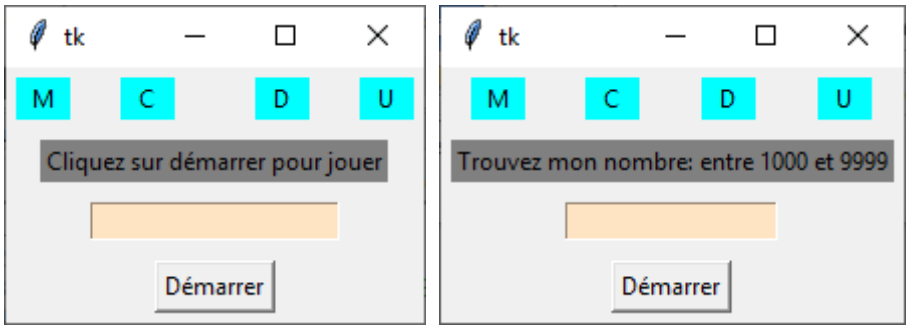
Labelordi = tk.Label(fen, **textvariable = TextOrdi** , bg ="grey")

```
Labelordi.grid(row=1,column=0, columnspan=4, padx = 5, pady = 5)
#-------------------------------------------------------------
demarrer=tk.Button(fen,text="Démarrer",command=lancerJeu)
demarrer.grid(row=3,column=1, columnspan=2, padx = 5, pady = 5)
#-------------------------------------------------------------
nombre = 0 #variable globale utilisée par la fonction lancerJeu
```

La fonction lancerJeu

```
from random import randint
def lancerJeu():
    global nombre
    nombre =str(randint(1000, 9999))
    TextOrdi.set("Trouvez mon nombre: entre 1000 et 9999")
```

Résultat : avant et après un clic sur « Démarrer »



La zone de texte et la touche <Return>

Nous ajoutons simplement après les instructions de création de ChampJoueur :

```
ChampJoueur.bind("<Return>",analyser)
```

Lorsque le joueur va appuyer sur la touche « **Entrée** », la fonction « **analyser** » va être exécutée.

C'est elle qui va déterminer ce qu'il faut afficher dans les labels correspondant aux milliers, centaines, dizaines et unités.

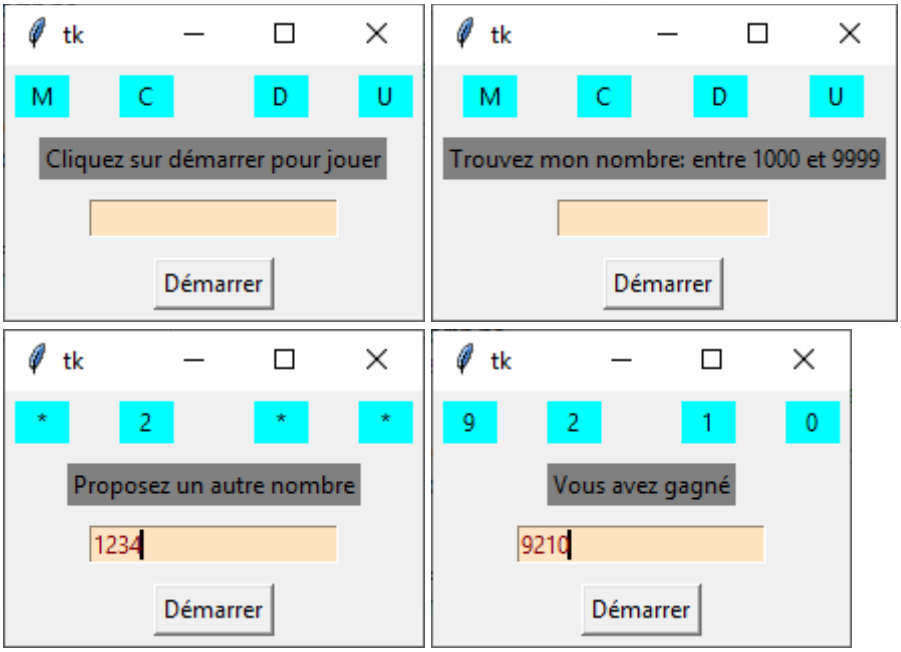
C’est elle qui doit aussi déterminer si le joueur a gagné.

S’il a gagné, la fonction affiche dans LabelOrdi : « Bravo, vous avez gagné », sinon elle affiche « Proposez un autre nombre »

Pour pouvoir modifier le texte dans les labels « chiffres », nous devons comme pour le LabelOrdi, créer pour chacun deux une variable Stringvar.

Enfin pour récupérer le nombre entré par le joueur, on utilise la fonction **get** sur le widget ChampJoueur. Voir les méthodes des zones de texte : http://tkinter.fdex.eu/doc/entw.html#Entry

Voici le programme complet à ce stade :




```
import tkinter as tk
from random import randint
def lancerJeu():
    global nombre
    nombre =str(randint(1000, 9999))
    TextOrdi.set("Trouvez mon nombre: entre 1000 et 9999")
```

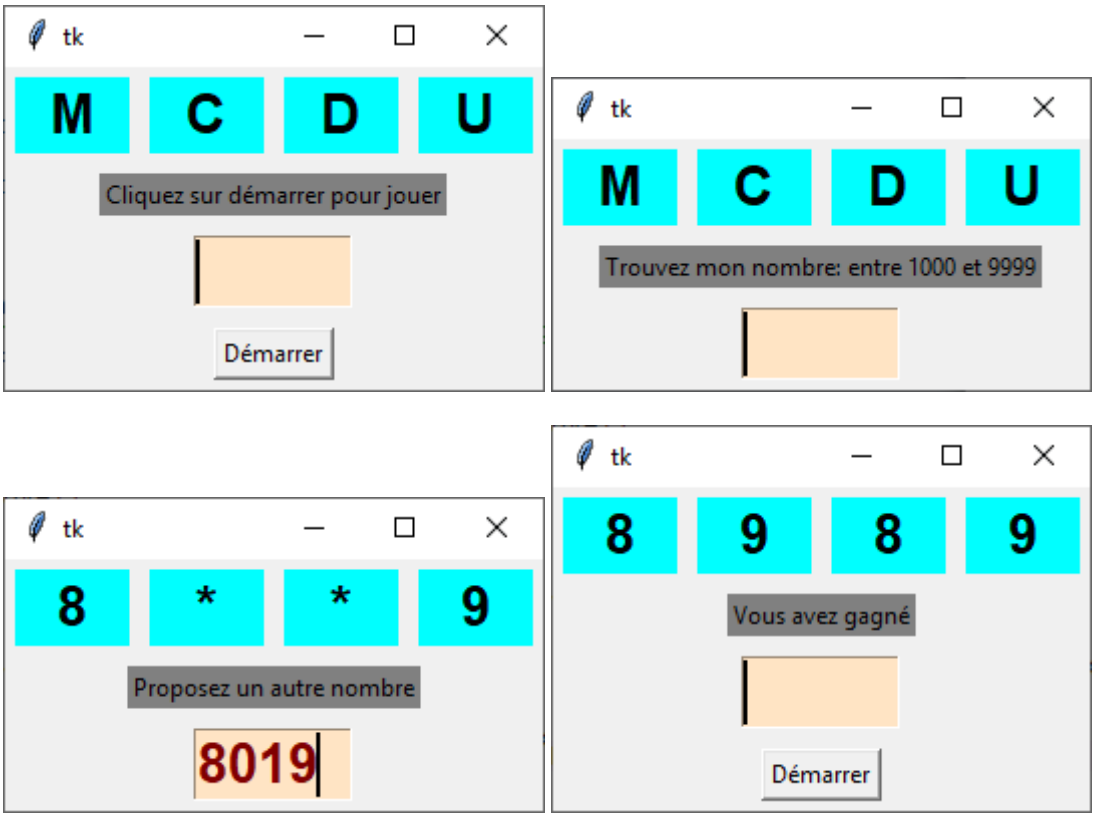
```
def analyser(event):
    global nombre
    chiffres = ["*","*","*","*"]
    trouve = 0
    nombre2 = ChampJoueur.get()
    print(nombre) #A enlever lorsqu'on joue vraiment
   for i in range (len(nombre)):
            if nombre[i] == nombre2[i]:
             chiffres[i] = nombre[i] #Bon chiffre à la bonne place
             trouve=trouve+1
  #affichage dans les labels chiffres
    TextM.set(chiffres[0])
    TextC.set(chiffres[1])
    TextD.set(chiffres[2])
    TextU.set(chiffres[3])
    if trouve==4:
        TextOrdi.set("Vous avez gagné")
    else:
        TextOrdi.set("Proposez un autre nombre")


#-----------------------------------------------------------------
#construction de la fenêtre
fen=tk.Tk()
# ----------------------------------------------------------------
TextM = tk.StringVar()
millier = tk.Label(fen,textvariable = TextM, bg="cyan", height=1, width = 3)
```

```
TextM.set('M')
TextC = tk.StringVar()
centaine = tk.Label(fen,textvariable = TextC, bg="cyan", height=1, width = 3)
TextC.set('C')
TextD = tk.StringVar()
dizaine = tk.Label(fen,textvariable = TextD, bg="cyan", height=1, width = 3)
TextD.set('D')
TextU = tk.StringVar()
unite = tk.Label(fen,textvariable = TextU, bg="cyan", height=1, width = 3)
TextU.set('U')
millier.grid(row = 0, column = 0, padx = 5, pady = 5)
centaine.grid(row = 0, column = 1,padx = 5, pady = 5)
dizaine.grid(row = 0, column = 2,padx = 5, pady = 5)
unite.grid(row = 0, column = 3,padx = 5, pady = 5)
#-------------------------------------------------------------
TextOrdi = tk.StringVar()
Labelordi = tk.Label(fen, textvariable = TextOrdi , bg ="grey")
Labelordi.grid(row=1,column=0, columnspan=4, padx = 5, pady = 5)
TextOrdi.set("Cliquez sur démarrer pour jouer")
#-------------------------------------------------------------
ChampJoueur = tk.Entry(fen,  bg ="bisque", fg="maroon")
ChampJoueur.grid(row=2,column=1,columnspan=2, padx = 5, pady = 5)
ChampJoueur.bind("<Return>",analyser)
#-------------------------------------------------------------
demarrer=tk.Button(fen,text="Démarrer",command=lancerJeu)
demarrer.grid(row=3,column=1, columnspan=2, padx = 5, pady = 5)
```

```
#-----------------------------------------------------------

nombre = 0

#fin boucle d'existence de la fenêtre : elle se termine lorsqu'on clique sur la croix en haut à #droite de la fenêtre

fen.mainloop()
```

Quelques améliorations :

- Changer la taille de la police dans les labels « chiffres » et le ChampJoueur pour que les chiffres soient plus gros.
- Lorsque le jeu a démarré, effacer le bouton « Démarrer » et le remettre lorsque la partie est terminée.
- Effacer le contenu de ChampJoueur pour redémarrer une nouvelle partie.



```
import tkinter as tk
from random import randint
import tkinter.font as tkFont
def lancerJeu():
    global nombre
    nombre =str(randint(1000, 9999))
    TextOrdi.set("Trouvez mon nombre: entre 1000 et 9999")
    demarrer.grid_forget() #supprime le bouton « demarrer »
    fen.update() #rafraichit la fenêtre

def analyser(event):
    global nombre
    chiffres = ["*","*","*","*"]
    trouve = 0
    nombre2 = ChampJoueur.get()
    for i in range (len(nombre)):
            if nombre[i] == nombre2[i]:
             chiffres[i] = nombre[i]
             trouve=trouve+1
    TextM.set(chiffres[0])
    TextC.set(chiffres[1])
    TextD.set(chiffres[2])
    TextU.set(chiffres[3])
```

```
    if trouve==4:
        TextOrdi.set("Vous avez gagné")
        #On efface le contenu de ChampJoueur
        #et on remet le bouton « Démarrer » en place
        Nombre1.set("")
        demarrer.grid(row=3,column=1, columnspan=2, padx = 5, pady = 5)
        fen.update()
    else:
        TextOrdi.set("Proposez un autre nombre")


# ----------------------------------------------------------------
#construction de la fenêtre
fen=tk.Tk()
police = tkFont.Font(family='Arial', size=20, weight='bold')
# ----------------------------------------------------------------
TextM = tk.StringVar()
millier = tk.Label(fen,textvariable = TextM, bg="cyan", height=1, width =
3,font=police)
TextM.set('M')
TextC = tk.StringVar()
centaine = tk.Label(fen,textvariable = TextC, bg="cyan", height=1, width =
3,font=police)
TextC.set('C')
TextD = tk.StringVar()
dizaine = tk.Label(fen,textvariable = TextD, bg="cyan", height=1, width =
3,font=police)
TextD.set('D')
```

```
TextU = tk.StringVar()
unite = tk.Label(fen,textvariable = TextU, bg="cyan", height=1, width = 3,font=police)
TextU.set('U')
millier.grid(row = 0, column = 0, padx = 5, pady = 5)
centaine.grid(row = 0, column = 1,padx = 5, pady = 5)
dizaine.grid(row = 0, column = 2,padx = 5, pady = 5)
unite.grid(row = 0, column = 3,padx = 5, pady = 5)
#-------------------------------------------------------
TextOrdi = tk.StringVar()
Labelordi = tk.Label(fen, textvariable = TextOrdi , bg ="grey")
Labelordi.grid(row=1,column=0, columnspan=4, padx = 5, pady = 5)
TextOrdi.set("Cliquez sur démarrer pour jouer")
#-------------------------------------------------------------
Nombre1= tk.IntVar()
Nombre1.set("")
ChampJoueur = tk.Entry(fen, textvariable= Nombre1, bg ="bisque",
fg="maroon", width="5",font=police)
ChampJoueur.focus_set()
ChampJoueur.grid(row=2,column=1,columnspan=2, padx = 5, pady = 5)
ChampJoueur.bind("<Return>",analyser)
#-------------------------------------------------------------
demarrer=tk.Button(fen,text="Démarrer",command=lancerJeu)
demarrer.grid(row=3,column=1,columnspan=2, padx = 5, pady = 5)
#-------------------------------------------------------
nombre = 0
#fin boucle d'existence de la fenêtre
fen.mainloop()
```