Jusqu'à présent, nous avons créé des programmes utilisant exclusivement la console pour l'interaction code-utilisateur. Cette interface est loin d'être idéale pour l'utilisateur.
Pour qu'un programme soit aisément utilisable, il faut prévoir une interface graphique : un programme qui affiche des fenêtres à l'écran pour expliquer à l'utilisateur ce qu'on attend de lui et afficher les résultats demandés, et qui s'exécute lorsqu'on lance le programme principal.

Nous utiliserons la bibliothèque tkinter de Python pour créer et gérer nos interfaces graphiques.
On installe la bibliothèque en tapant pip install tkinter dans la console.
On charge ensuite cette bibliothèque en entrant : from tkinter import *

Partie I : Introduction à tkinter

Pour créer un logiciel graphique, il faut tout d'abord créer une fenêtre.
On ajoute ensuite dans une fenêtre des éléments graphiques que l'on nomme widget. Quelques exemples de widget :

- *Button* : proposer une action à l'utilisateur.
- *Label* : espaces prévus pour écrire du texte.
- *Canvas* : espace dans lequel vous pouvez dessiner ou écrire
- *Frame* : cadres qui permettent de séparer des éléments.

A chaque widget est une fonction qui demande arguments spécifiques, ainsi que des options (taille, couleur, etc…).
Dans chaque cas, **on assigne une variable au widget**, afin de le repérer, et on le « compile » dans la fenêtre. Il faut enfin afficher la fenêtre.

Fonction	Description
fenetre = *Tk()*	Créer une fenêtre
bouton = *Button(***fenetre***, **text** =* "Texte à afficher"*, **command** =* Fonction*)*	Créer un bouton
label = *Label(***fenetre***, **text** =* "Texte à afficher"*)*	Créer un label
canevas = *Canvas(***fenetre**, ***width*****=** Taille en px, ***height*****=** Taille en px*)*	Créer un canevas
cadre = *Frame(***fenetre***)*	Créer un cadre
NomWidget.*pack()*	Ajouter le widget
NomFenetre.*mainloop()*	Afficher la fenêtre

Exemple : Tester le code suivant:

```
1 from tkinter import *
2 fen=Tk()          #on crée une fenêtre nommée fen1
3 tex = Label(fen, text = 'Bonjour à tous !',fg = 'red')          #on crée du texte pour la fenêtre
4 tex.pack()        #on ajoute le texte dans la fenêtre
5 can = Canvas(fen, width =100, height =100, bg = 'LightGreen' )     #on crée un canevas pour la fenêtre
6 can.pack()        #on ajoute le canevas dans la fenêtre
7 bou = Button(fen, text='Quitter', command = fen.destroy) #on crée un bouton Quitter pour la fenêtre.
8 bou.pack()        #on ajoute le bouton dans la fenêtre
9 fen.mainloop()          #on affiche la fenêtre
```

I.1. Donner 2 exemples d'option que l'on peut ajouter aux fonctions des widgets telles que *Label* et *Canvas*. Que font ces options ? Quelle est la syntaxe à utiliser ?

I.2. Donner un exemple de commande que l'on peut ajouter à la fonction *Button*. Que fait cette commande ? Quelle est la syntaxe à utiliser ?

I.3. Retirer les options des fonctions *Label* et *Canvas*. Quelles sont alors les valeurs de base des options ?

I.4. Modifier la ligne 8 en utilisant bou.pack(side=RIGHT) à la place. A quoi sert l'option *side* ?

I.5. Sans toucher à la ligne 8, modifier la ligne 6 en utilisant can.pack(side=LEFT) . Que peut-on en conclure ?

I.6. Supprimer la ligne 8 dans le code. Expliquer ce qui se passe.

Il existe différentes fonctions pour dessiner dans un canevas. Il faut alors préciser dans quel canevas faire le dessin en début du nom de la fonction.
Quelques fonctions sont listées ci-dessous :

Fonction	Description
canevas.*create_line(*x1, y1, x2,y2*)*	Dessiner une ligne entre le point de coordonnées (x_1, y_1) et le point de coordonnées (x_2, y_2)
canevas.*create_arc(*x1, y1, x2,y2*)*	Dessiner un arc entre le point de coordonnées (x_1, y_1) et le point de coordonnées (x_2, y_2)
canevas.*create_rectangle(*x1, y1, x2,y2*)*	Dessiner un rectangle défini par 2 coins donnés par les points de coordonnées (x_1, y_1) et (x_2, y_2)
canevas.*create_oval(*x1, y1, x2,y2*)*	Dessiner un cercle dans le rectangle défini par 2 coins donnés par les points de coordonnées (x_1, y_1) et (x_2, y_2)

Exemple : Tester le code suivant:

```
1  from tkinter import *
2  fen = Tk()
3  can = Canvas(fen, width =200, height =200, bg = 'ivory' )
4  can.pack(side =TOP, padx =5, pady =5)
5  rayon = 90
6  c = 0
7  while rayon > 0:
8      if c%2 == 0:
9          col="red"
10     else:
11         col="blue"
12     can.create_oval(100-rayon, 100-rayon, 100+rayon, 100+rayon, outline='black', fill=col)
13     rayon -= 15
14     c += 1
15 can.create_line(100, 0, 100, 200, width=3, fill = 'green' )
16 can.create_line(0, 100, 200, 100, width=3, fill = 'green' )
17 fen.mainloop()
```

I.7. Commenter ce code.

I.8. Quelles sont les différentes options utilisées dans les widgets ?

Il peut aussi être utile de créer un champ de saisie dans une fenêtre, afin que l'utilisateur puisse interagir avec le programme. Pour cela, on utilise un nouveau constructeur de widget : *Entry*.

Fonction	Description
entree = *Entry(***fenetre***)*	Fenêtre de saisie. Le résultat de la saisie sera enregistré dans une variable nommé *event*

Exemple : Tester le code suivant:

```
1  from tkinter import *
2  def evaluer(event):        #fonction permettant de faire le calcul
3      # la méthode get() permet de récupérer ce qui est saisi, sous forme de chaîne de caractères
4      chaine.configure(text = "Résultat = " + str(eval(entree.get())))
5
6  fenetre = Tk()
7  entree = Entry(fenetre)            #créer une fenêtre de saisie
8  entree.bind("<Return>", evaluer)          #associer une action à la fenêtre de saisie
9  chaine = Label(fenetre)            #créer un label
10 entree.pack()
11 chaine.pack()
12 fenetre.mainloop()
```

I.9. Expliquer ce qui se passe lors de l'exécution de la ligne 8 du code.

I.10. Dans la fonction *evaluer*, dans quel type de variable est donné le résultat ? Pourquoi utiliser ce type de variable ?

Les actions utilisateurs (clic de souris, touche de clavier appuyée…) peuvent être récupérées à travers les évènements, donnés dans la variable *event***.**
Cette variable est **obligatoire en seul argument*** **lorsque l'on définit une fonction gestionnaire d'évènements qui est associée à un widget par la méthode *bind***. On doit alors l'utiliser comme premier argument. Cet argument désigne un objet créé automatiquement par tkinter, qui permet de transmettre au gestionnaire d'événement un certain nombre d'attributs de l'événement :

- le type d'événement : déplacement de la souris, enfoncement ou relâchement de l'un de ses boutons, appui sur une touche du clavier, entrée du curseur dans une zone prédéfinie, ouverture ou fermeture d'une fenêtre, etc…
- une série de propriétés de l'événement : l'instant où il s'est produit, ses coordonnées, les caractéristiques du ou des widget(s) concerné(s), etc…

* Sauf en utilisant la méthode *lambda* – voir documentation sur internet

Une liste non-exhaustive des différents évènements est donnée ci-dessous :

Code	Evenement
<Button-1>	Clic gauche
<Button-2>	Clic milieu
<Button-3>	Clic droit
<Double-Button-1>	Double clic gauche
<KeyPress>	Pression sur une touche
<KeyPress-a>	Pression sur la touche a minuscule
<KeyPress-A>	Pression sur la touche A majuscule
<Return>	Pression sur la touche Entrer
<Up>	Pression sur la touche flèche vers le haut
<Down>	Pression sur la touche flèche vers le bas

Pour plus d'information, de l'aide sur les différentes fonctions de *tkinter* est disponible sur :
http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Tournez la page pour la partie II

Partie II : A vous de jouer !

II.1. On considère le script lignes.py joint en fichier.

a. Essayer de comprendre ce qu'il fait avant de l'essayer (en commentant le code), puis essayer ce code.

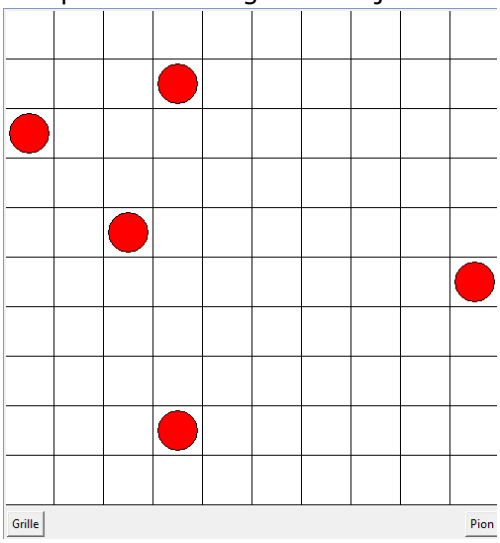
b. Pourquoi utilise-t-on des variables globales ?

c. Modifier ce programme afin qu'il effectue les tâches suivantes :

- Les lignes sont désormais horizontales et parallèles.
- Agrandir le canevas pour qu'il ait une largeur de 500 unités et une hauteur de 650 pixels. Modifier la taille des lignes en conséquence.
- Ajouter une fonction *dessineViseur()* qui trace deux lignes rouges en croix au centre du canevas : l'une horizontale et l'autre verticale.
  Ajouter aussi un bouton viseur qui provoque l'affichage de ces croix.
- Remplacer *create_line* par *create_rectangle*, puis *create_arc* et *create_oval*.

II.2. Écrire un programme qui crée une fenêtre qui contient six boutons. Les cinq premiers boutons permettent chacun l'affichage d'un cercle de couleur représentant l'un des anneaux olympique. Le dernier bouton permet de fermer la fenêtre.

II.3. Écrire un programme qui, sur un canevas de 500×500, fait apparaître une grille de 10 cases par 10 cases sur pression d'un bouton, puis fait apparaître un pion aléatoirement sur l'une des cases en pressant un second bouton (mais uniquement si la grille a déjà été créée).



II.4. Le code clic.py donne un deuxième exemple de programmation par évènement.
En s'aidant de ce code, écrire un nouveau programme qui ouvre une fenêtre et affiche un cercle rouge à l'endroit où l'utilisateur à fait un clic droit et un carré bleu là où il a fait un clic gauche.