Python objet – TP 4

Documentation générale (en français) : http://tkinter.fdex.eu/doc/caw.html

1. Prise en main de TkInter

Ouvrez le fichier ***tkinter_base.py***. Il décrit la structure globale d'une application Tkinter.

Quelques remarques :

- L'option command lors de la création du bouton permet de lier le bouton à une méthode ou une fonction. Vous remarquerez que dans l'exemple, l'option vaut : command = fen.quit , sans parenthèses à la fin. C'est parce que l'option command prend en paramètre un « callback », c'est-à-dire un « objet fonction », et non pas l'appel à une fonction. On verra plus tard comment utiliser cela quand on veut y passer certains paramètres spécifiques.
- On utilisera la méthode ***pack()*** pour disposer les boutons. On pourrait également utiliser le gestionnaire de positionnement basé sur la méthode ***grid()***. La documentation de ***grid()*** est ici : http://tkinter.fdex.eu/doc/gp.html

2. Dessiner des formes simples / faire des boutons

Lignes droites

Pour ajouter une ligne droite dans une fenêtre Tkinter, on utilise la méthode ***create_line()*** du canevas principal. Cette méthode a la syntaxe suivante :

`Canvas.create_line`(*x0*, *y0*, *x1*, *y1*, ..., *xn*, *yn*, *option*, ...)

Les coordonnées *x0*, *y0*, *x1*, *y1*, ..., *xn*, *yn* sont les coordonnées des points à relier.
Par exemple, pour une ligne simple entre les points (34,5) et (75,42), on appelle la méthode **Canvas.create_line(34,5,75,42).**

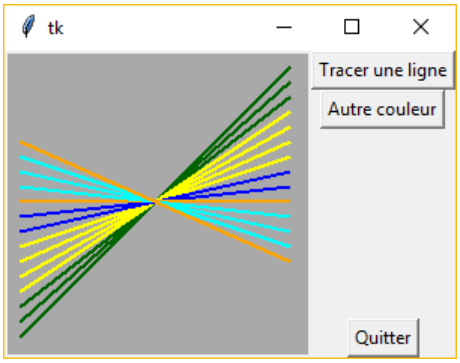
ATTENTION : dans le canvas, le point (0,0) est en haut à gauche !!!

Voyez la section http://tkinter.fdex.eu/doc/caw.html#Canvas.create_line pour comprendre les différentes options possibles lors du dessin de la ligne. Les options les plus importantes sont :

- ***fill*** : la couleur de la ligne. Tkinter connait de nombreuses couleurs par défaut (avec leur nom en anglais), voir ici : http://cs111.wellesley.edu/~cs111/archive/cs111_spring15/public_html/labs/lab12/tkintercolor.html
- ***width*** : l'épaisseur de la ligne, en pixels par défaut
- ***arrow*** : Par défaut, la ligne n'est pas terminée par une flèche. Utiliser `arrow='first'` pour obtenir une flèche au point *(x0, y0)* de la ligne. Utilisez `arrow='last'` pour obtenir une flèche à l'autre extrémité. Utilisez `arrow='both'` pour en avoir à chaque extrémité.
- ***dash*** : pour dessiner des pointillés. Par défaut, les lignes sont pleines. L'option `dash=(3, 5)` produit une ligne où le parties tracées font 3 pixels et où les parties vides en font 5. `dash=(7, 1, 1, 1)` produirait un motif de base où les partie tracées mesureraient 7 puis 1 pixels séparés par des parties vides de 1 pixel.

Complétez le fichier ***tkinter_lignes.py*** de la façon suivante :

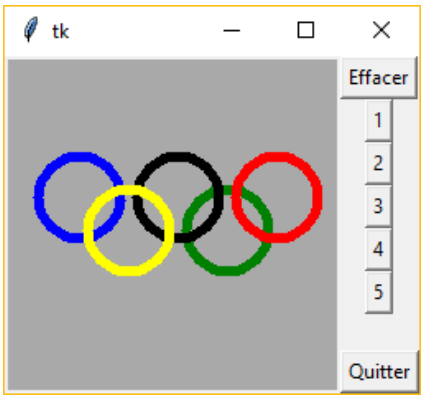
- Complétez la ligne 12 pour tracer une ligne en utilisant la fonction create_line()
- Complétez la méthode ***changeStyle()*** pour qu'elle change la couleur de la ligne dessinée, et le style de la ligne
- Ajoutez un bouton qui appellera la méthode ***changeStyle()***



Le programme doit pouvoir générer une fenêtre qui ressemble à ça :

Anneaux olympiques

Le but de cette partie est de créer une interface graphique qui permet de dessiner plusieurs anneaux de différentes couleurs.



La méthode permettant de dessiner un ovale (et donc un cercle) est la suivante :

```
canvas.create_oval(x0, y0, x1, y1, options)
```

Les coordonnées (x0,y0) et (x1,y1) sont les coordonnées des coins « haut gauche » et « bas droite » du rectangle qui contient l'ovale. Si x1-x0 = y1-y0, alors l'ovale devient un carré.

Ouvrez le fichier tkinter_anneaux.py. Si vous le lancez, vous verrez que le bouton « Tracer » dessine un cercle bleu de 6 pixels d'épaisseur. La méthode ***clear()*** efface tout dessin du canevas, grâce à la méthode ***delete()*** de Canvas.

Modifier le fichier pour que l'interface graphique ressemble à l'image ci-dessus (chaque bouton 1 à 5 affiche un des anneaux, et le bouton Effacer les efface tous) :

- Modifiez la méthode ***drawRing()*** pour qu'elle affiche 5 anneaux avec leurs couleurs et positions correctes. Pour information, les anneaux ont un diamètre de 50 pixels, un écart horizontal de 30 pixels et un écart vertical de 20 pixels.
- Ajoutez un bouton qui déclenche le dessin de chaque anneau. Attention : On aurait aimé écrire ***command = d.drawRing(x,y,couleur),*** mais l'option command ne prend pas en paramètre un appel de méthode mais seulement une méthode. Pour passer des paramètres, vous devez suivre la syntaxe suivante :

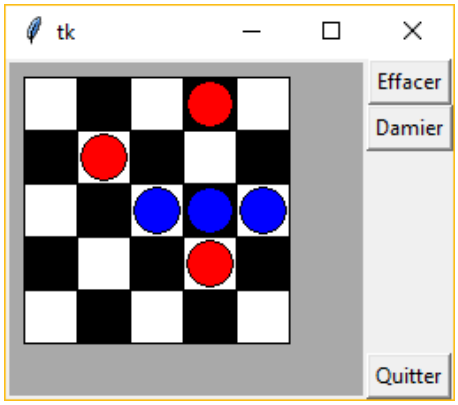
  command = lambda : d.drawRing(x,y,couleur)

Dessiner des formes complexes et gérer l'interaction à la souris

Nous allons maintenant dessiner un damier sur lequel nous pourrons jouer. Ouvrez le fichier ***tkinter_damier.py***. Pour l'instant, il ne se passe pas grand chose. Mais quand vous cliquez avec le bouton gauche de la souris, un petit cercle blanc apparaît. C'est grâce à la méthode ***Canvas.bind ("<Button-1>", d.pointeur)*** qui crée un lien entre un « événement », c'est-à-dire une touche enfoncée, un clic, ou autre chose, et le lancement d'une fonction, ici la méthode ***pointeur()*** de la classe Damier.

Plus de détails sur la gestion des événements ici : http://tkinter.fdex.eu/doc/event.html

- Créez une grille de 5x5 cases
- Créez un damier de 25 cases avec des cases noires et blanches (la case en haut à gauche doit être blanche)
- Ajoutez un bouton Effacer qui efface tous les dessins présents sur la fenêtre.
- Ajoutez un bouton Damier qui dessine un damier.
- Faites en sorte qu'au clic de la souris sur une case, on fait apparaitre un pion (un cercle blanc) dans la case.



- On veut maintenant que l'utilisateur ne puisse pas ajouter de pion sur une case déjà occupée. Vous aurez besoin de rajouter (si ce n'est pas déjà le cas) une gestion de votre damier case par case, pour pouvoir récupérer l'état d'une certaine case.
- Faites en sorte que les couleurs des pions posés soient alternativement bleus et rouges.
- Lors de l'ajout d'un pion, vérifiez si la partie est gagnée par un des joueurs ou pas. On considère que la partie est gagnée si trois pions de la même couleur sont alignés (en horizontal, vertical ou diagonal).

3. Animation / Animation automatique

Ouvrez le fichier ***tkinter_balle.py.*** Lancez-le : il affiche une balle rouge qui tourne autour du cadre. L'animation est principalement dépendante de deux fonctions :

- La méthode ***Canvas.coords(obj,x1,y1,x2,y2)*** permet de mettre à jour les coordonnées d'un objet déjà dessiné et de le redessiner à ces nouvelles coordonnées (si on faisait seulement Canvas.create_oval(...) on aurait une multitude de cercles dessinés superposés !).
- La méthode ***Tk.after(time, method)*** signifie : après la durée time, exécuter la méthode method. Comme ici on lui donne en paramètre la méthode move(), cela fait boucler le processus. Comme pour l'option command vue plus tôt, le paramètre method est un callback, donc un nom de méthode, pas un appel !

Vous allez maintenant modifier le programme :

- Modifiez le programme pour que la balle change de couleur à chaque quart de tour. Renseignez-vous sur la méthode ***Canvas.itemconfigure()*** pour cela.
- Modifiez le programme pour que la balle rapetisse quand on clique dessus avec le bouton gauche de la souris. Vous aurez besoin de créer de nouvelles variables et méthodes.