

## TP de Langages de script n° 2 : Introduction à Tkinter et aux listes

### Exercice 1 : Premier programme graphique.

1. Dans un interpréteur python, importez tout le contenu du module `Tkinter`.
2. Ce module contient une classe `Tk`. Instanciez un objet `fenetre` de cette classe grâce à la commande `fenetre = Tk()`. Vous voyez apparaître une fenêtre<sup>1</sup>.
3. Regardez l'aide en ligne de la commande `__init__` de la classe `Tk`. Le premier argument (`self`) désigne l'objet à travers lequel la méthode est invoquée. Son nom est une convention, mais le premier argument d'une méthode représente toujours cet objet.
4. Nous allons commencer par construire un bouton pour fermer la fenêtre. Pour cela il faut créer une instance de la classe `Button` et la placer dans la fenêtre.
  - Tapez la commande
 

```
quitter = Button(fenetre, text="Quitter", command=fenetre.destroy)
```
  - Le bouton précédemment créé existe mais il n'est pas encore visible. Pour cela il faut le placer dans la fenêtre. Il y a plusieurs façons de faire, la plus simple est d'invoquer la méthode `pack` sans argument. Placez le bouton `quitter`.
 Notez que si on n'a pas besoin de faire référence au bouton dans la suite du code, on peut écrire directement
 

```
Button(fenetre, text="Quitter", command=fenetre.destroy).pack()
```
5. Cliquez sur le bouton précédemment : selon les cas, la fenêtre se ferme ou rien ne se passe. Si rien ne se passe, pour que le bouton soit actif, mettez la fenêtre en attente d'événements grâce à la méthode `mainloop`.

Vous aurez noté que la méthode `mainloop` ne prend pas la main avant la destruction de la fenêtre, il faut donc bien tout mettre en place *avant* de lancer `mainloop`<sup>2</sup>.

### Exercice 2 : Mandala

Nous allons créer un mandala tout simple, comme illustré à la figure 1.

La méthode `create_oval` de la classe `Tkinter.Canvas` s'utilise pour tracer un ellipse de la façon suivante :

```
create_oval(x_NO, y_NO, x_SE, y_SE)
```

Les arguments représentent les abscisses et ordonnées des extrémités NO et SE du rectangle dans lequel l'ellipse sera inscrite. Si l'ellipse est inscrite dans un carré, c'est un cercle.

1. Ouvrez une fenêtre `Tk`. Créez et placez un canvas (class `Canvas`) dans cette fenêtre (un `canvas` est un élément graphique dans lequel nous allons pouvoir dessiner).
2. Tracez un premier cercle centré dans votre fenêtre. Nous appellerons ce cercle `C` dans la suite.

<sup>1</sup>A noter que cela peut dépendre de la version de python; éventuellement la fenêtre n'apparaît qu'après l'invocation de la méthode `mainloop` (cf. question 5).

<sup>2</sup>Il existe d'autres interpréteurs python, comme `ipython` qui permettent d'éviter cet inconvénient.

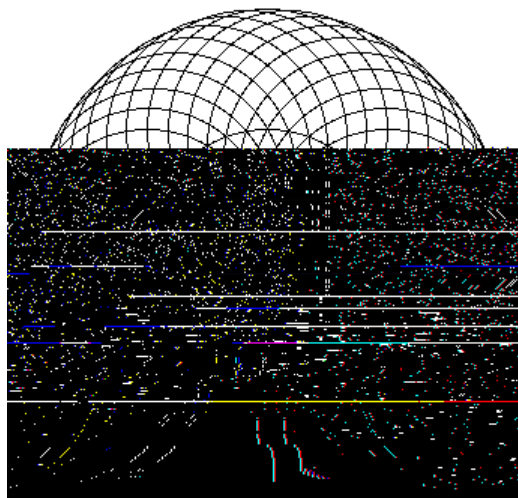


FIG. 1 – Un mandala tout simpl

- On rappelle qu' sur un cercle de centre  $(x, y)$  et de rayon  $r$ , les coordonnées du point situé à l'angle  $\alpha$  sont données par  $(x + r \cos \alpha, y + r \sin \alpha)$ . En faisant varier  $\alpha$  de 0 à 360 par pas de 10, dessinez les cercles centrés sur  $\mathcal{C}$  et régulièrement disposés.

**Attention :** en Python, les angles sont exprimés par défaut en radians ! Pensez aux conversions.

- Faites varier les changements du paramètre  $\alpha$  (différents pas, pas variables, etc.) pour obtenir différents dessins.

### Exercice 3 : Listes - manipulations élémentaires.

Un list python (typ `list`) est une collection ordonnée d'éléments non nécessairement du même type. Elle est délimitée par des crochets `[ ]` et ses éléments sont séparés par des virgules. Par exemple `[0, 's', Tk()]` est une list ; `[[1, 2, 3], 1, [1]]` est aussi une list.

Comme les chaînes de caractères, les listes font partie de ce qu'on appelle les *séquences* en python. Elles sont un outil utile et puissant comme nous le verrons dans les semaines futures.

Ce que nous avons vu sur les chaînes de caractères s'applique également aux listes : extraction d'un élément, d'une suite contiguë d'éléments, signification du mot-clé `in`, etc.

- Soit `L` une list. Que se passe-t-il si on affiche un valeur à `L[1:3]` ? à `L[1:]` ? à `L[:]` ?
- Ecrivez une fonction `inversion` qui demande à l'utilisateur de saisir une ligne de texte et renvoie une list des mots dans l'ordre inversé (on supposera sans faire de vérification que l'utilisateur n'a saisi que des lettres ou des espaces).
- Ecrivez une fonction `sans_e` qui prend en argument une list de mots et affiche la list de ceux qui ne contiennent pas la lettre 'e'.

### Exercice 4 : Listes - un peu de complexité.

- Ecrivez une fonction `inclus` qui prend en argument deux listes et dit si les éléments de la première list apparaissent dans la deuxième, dans le même ordre.

2. Ecrivez une fonction `nti_begue` qui prend en argument un chaîne de caractères et crée un nouveau dans laquelle ont été supprimés les mots consécutifs identiques. Votre fonction devra transformer le texte d'origine en liste de mots avant tout traitement et transformer la liste nouvelle ment obtenue en chaîne de caractères. Pour cela, utilisez l'aide en ligne des méthodes `split` et `join` du type `str`.
3. On considère une liste de listes représentant une image en noir et blanc (sans niveau de gris) qui a subi une compression élémentaire en tant que comptés des cas adjacents identiques. On veut décompresser une telle image. Ainsi à partir de l'image compressée donnée par la liste

```
[[ 'N', 4, 'B'],  
 [ 'B', 'N', 3, 'B'],  
 [ 'N', 'B', 4]]
```

on obtiendra l'image non compressée donnée par la liste

```
[[ 'N', 'N', 'N', 'N', 'B'],  
 [ 'B', 'N', 'N', 'N', 'B'],  
 [ 'N', 'B', 'B', 'B', 'B']]
```

Ecrivez la fonction `decompresser`, puis, si vous avez le temps, une fonction d'affichage de l'image noir et blanc utilisant `Tkinter`. Attention : ce n'est pas la bonne façon en python.

### Exercice 5 : Un aquarium.

Téléchargez le fichier `quarium.py`. Vous pouvez regarder la documentation du module `quarium` grâce à `pydoc`.

On vous demande de construire, en vous servant du module `quarium`, un aquarium contenant plusieurs poissons de divers couleurs qui font chacun des aller-retour dans l'aquarium.