TP de Langages de script n° 3 : Introd. aux listes et à `Tkinter`

Exercice 1 : Listes.

Une liste python (type `list`) est une collection ordonnée d'éléments non nécessairement de mêmes natures. Elle est délimitée par des crochets (`[ ]`) et ses éléments sont séparés par des virgules. Par exemple `[0, 's', tkinter.Tk()]` est une liste ; `[[1, 2, 3], 1, [1]]` est aussi une liste.

Comme les chaînes de caractères, les listes font partie de ce que l'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-clef `in`, etc.

1. Soit `L` une liste. Que se passe-t-il si on affecte une valeur à `L[1:3]` ? à `L[1:]` ? à `L[:]` ?
2. Ecrivez une fonction `inversion` qui demande à l'utilisateur de saisir une ligne de texte et renvoie une liste des mots dans l'ordre inverse (on supposera sans faire de vérification que l'utilisateur ne saisit que des lettres ou des espaces). Pour découper une chaine de caractères en une liste de mots, vous pourrez utiliser la méthode split.
3. Ecrivez une fonction `sans_e` qui prend en argument une liste de mots et affiche la liste de ceux qui ne contiennent pas la lettre `'e'`.
4. Ecrivez une fonction `inclus` qui prend en argument deux listes d'entiers et dit si les éléments de la première liste apparaissent dans la deuxième, dans le même ordre.
5. Ecrivez une fonction `anti_begue` qui prend en argument une chaîne de caractères et en crée une nouvelle 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 nouvellement obtenue en chaîne de caractères. Pour cela, regarder l'aide en ligne des méthodes `split` et `join` du type `str`.

Exercice 2 : Premier programme graphique.

1. Dans un interpréteur python, importez le module `tkinter`.
2. Ce module contient une classe `Tk`. Instanciez un objet `fenetre` de cette classe grâce à la commande `fenetre = tkinter.Tk()`. Vous voyez apparaître une fenêtre [1].
3. Regardez l'aide en ligne de la commande `__init__` de la classe `tkinter.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 = tkinter.Button(fenetre, text="Quitter", command=fenetre.destroy)`
   - Le bouton précédement créé existe mais il n'est pas encore visible. Pour cela il faut le placer dans la fenêtre. Il y a plusieurs façon 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

   ```
   tkinter.Button(fenetre, text="Quitter", command=fenetre.destroy).pack()
   ```
5. Cliquez sur le bouton précédent : 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 rend pas la main avant la destruction de la fenêtre, il faut donc bien tout mettre en place *avant* de lancer `tkinter.mainloop` [2].

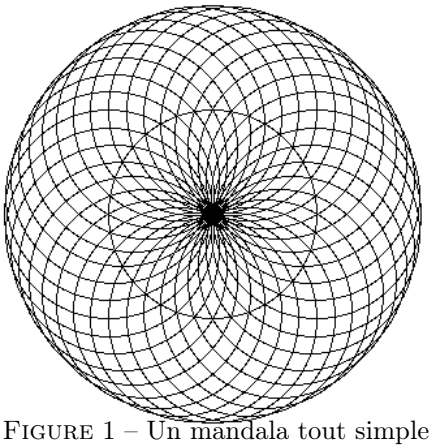

FIGURE 1 – Un mandala tout simple

Exercice 3 : 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 une ellipse de cette façon :

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

Les arguments représentent les abscisses et ordonnées des extrêmités 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 à l'aide de la classe `Tk`. Créez et placez un canvas (classe `Canvas`) dans cette fenètre (un *canvas* est un élément graphique dans lequel nous allons pouvoir dessiner). Avec tkinter, les coordonnées (0,0) se trouvent en haut à gauche.
2. Tracez un premier cercle centré dans votre fenêtre. Nous appellerons ce cercle $\mathcal{C}$ dans la suite.
3. On rappelle que sur un cercle de centre (x, y) et de rayon r, les coordonnées du point situé à l'angle α 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 ! Penser aux conversions.
4. Faites varier les changements du paramètre α (différents pas, pas variables, etc.) pour obtenir différents dessins.

Exercice 4 : Listes + tkinter

On considère une liste de listes représentant une image en noir et blanc (sans niveau de gris) qui a subi une compression élémentaire en tenant compte des cases adjacentes 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`.

Exercice 5 : Un aquarium.

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

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

1. A noter que cela peut dépendre de la version de python ; éventuellement la fenêtre n'apparaît qu'après l'invocation de la méthode `tkinter.mainloop` (cf. question 5).
2. Il existe d'autres interpréteurs python, comme `ipython`, qui permettent d'éviter cet inconvénient.