TD : Les listes en pratique

L'objectif de ce TD est vous familiariser avec la notion de liste.

1) Création de listes

Question 1. Créer une fonction qui renvoie une liste contenant les `n` premiers nombres entiers positifs (à partir de 0), où `n` est l'argument de la fonction.

Remarque : En Python, le même type `list` est utilisé pour représenter les tableaux et les listes. On pourrait donc répondre à cette question en créant un tableau de `n` et en remplissant ensuite les cases (comme on l'a fait dans la TD précédent).

Cependant, l'idée est ici de vous faire manipuler les listes et les fonctions associées (en particulier `append`, `pop` et la liste vide `[]`). Sauf indication contraire, vous n'avez donc pas le droit de créer des tableaux non vides (via la syntaxe `nom = [valeur] * taille`) pour les exercices de ce TD.

Question 2. Créer une fonction qui renvoie la liste des multiples de 11 compris entre `a` et `b` (inclus), où `a` et `b` sont deux nombres entiers positifs passés en argument à la fonction.

2) Diviseurs

Question 3. Ecrire une fonction qui renvoie la liste des diviseurs d'un nombre entier positif.

Rappel : a est un diviseur de n si et seulement si il existe un nombre entier k (strictement positif) tel que $a \times k = n$

Question 4. Ecrire une fonction qui détermine sur un nombre entier est premier

Rappel : Un nombre premier a exactement deux diviseurs (distincts) : 1 et lui-même

Question 5. Créer une fonction qui renvoie la liste des nombres premiers compris entre `a` et `b` (inclus), où `a` et `b` sont deux nombres entiers positifs passés en argument à la fonction.

Question 6. Créer une fonction qui calcule le plus grand diviseur commun de deux nombres entiers.

3) Recherche, insertion et suppression

Question 7. Ecrire une fonction qui prend en argument une liste `l` (qu'on suppose non triée) et une valeur `x` et qui détermine (en renvoyant un booléen) si l'élément `x` apparaît dans la liste `l`.

Remarque : Attention, la liste initiale ne doit pas être modifiée.

Question 8. Ecrire une fonction qui prend en argument une liste `l` (qu'on suppose triée) et une valeur `x` et qui détermine (en renvoyant un booléen) si l'élément `x` apparaît dans la liste `l`.

Remarque : Cette fonction doit tirer parti du fait que la liste est triée

Question 9. Ecrire une fonction qui prend en argument une liste `l` (qu'on suppose triée) et une valeur `x`, et qui insère l'élément `x` à sa place dans la liste `l`.

Remarque : Cette fois, la liste initiale doit bien être modifiée.

Question 10. Ecrire une fonction qui prend en argument une liste `l` (qu'on suppose non triée) et une valeur `x`, et qui supprime toutes les occurrences de l'élément `x` dans la liste `l`.

4) Les tours d'Hanoï

a. Présentation du problème

Le problème des Tours d'Hanoï consiste à faire passer des disques d'un pylône (en général celui de gauche) vers un autre (en général celui de droite).

Les règles sont simples :

- On ne déplace qu'un disque à la fois (sinon il n'y a plus de problème)
- Il est interdit de placer un disque sur un autre disque de diamètre plus petit (sur le schéma ci-dessus, le disque au milieu ne peut pas aller sur le pylône de droite)

b. Avec du papier

Question 11. Pour commencer, reproduire le problème avec de simples bouts de papier :

- Prenez une feuille de brouillon
- Découpez-la en 4 morceaux de tailles bien différentes
- Définissez 3 emplacements sur votre table
- Empilez les 4 bouts de papier du plus large au plus petit sur l'emplacement 1
- Essayez de faire passer ces 4 bouts de papier sur l'emplacement 3 en les déplaçant un par un et sans jamais mettre un bout de papier sur un morceau plus petit

c. Récupération du fichier source

Localisez le fichier source disponible sur le support en ligne du cours et enregistrez-le dans votre répertoire de travail (votre fichier de travail et ce fichier source doivent être au même endroit).

Pour pouvoir utiliser les éléments déclarés dans le fichier source, tapez les lignes suivantes au début de votre fichier de travail :

```
import hanoi
hanoi = hanoi.HanoiGUI()
```

d. Représentation des données

On va représenter Les tours d'Hanoï sont représentées par trois références sur des listes d'entiers, nommées `tour1`, `tour2` et `tour3`. Dans chaque liste, chaque entier correspond à un disque (ce disque étant d'autant plus large que l'entier est grand)

Par exemple, pour le schéma ci-contre, on a :

- `tour1 = [8, 7, 6, 5, 4]`
- `tour2 = [3]`
- `tour3 = [2, 1]`

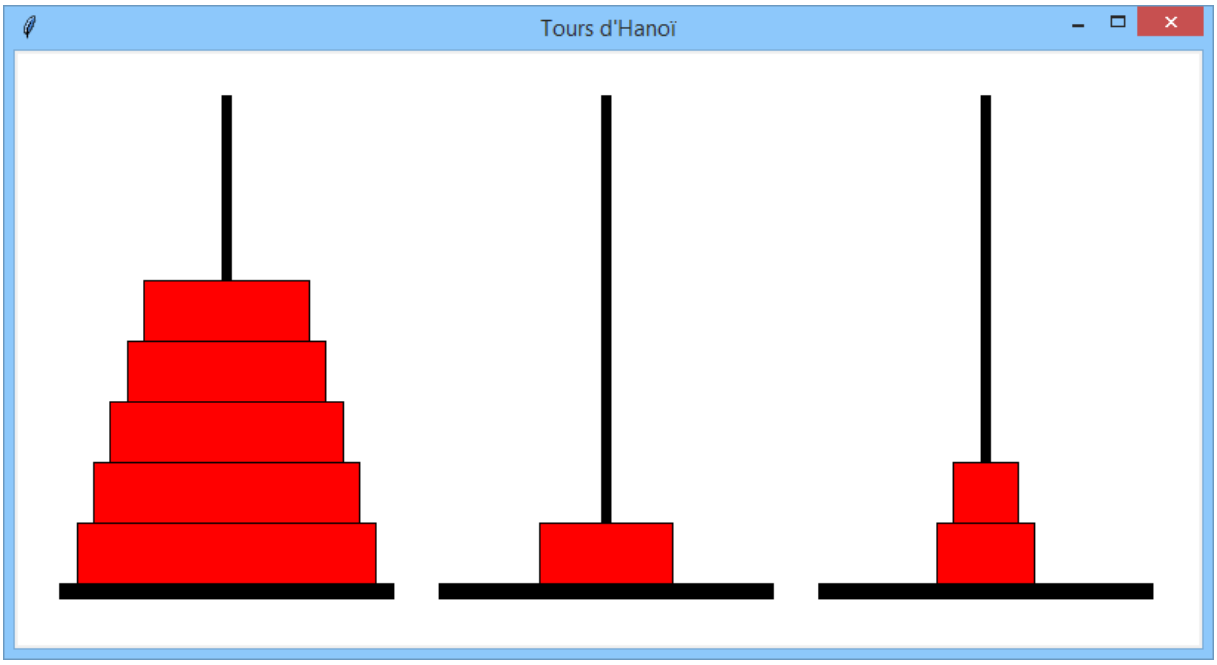
L'objectif de ce TP va être d'écrire des fonctions qui manipuleront ces trois listes afin de résoudre le problème des tours d'Hanoï.

Question 12. Ecrire une fonction qui prend en argument un nombre entier positif `n` et qui renvoie trois listes (sous forme d'un triplet) correspondant à la position de départ pour `n` disques.

Exemple : Pour 4 disques, cette fonction doit renvoyer `([4, 3, 2, 1], [], [])`

La fonction `print` vous permet de tester directement le contenu de vos listes. Cependant, pour un rendu plus graphique, vous pouvez utiliser la fonction `hanoi.afficher` disponible dans le fichier source.

Son utilisation est simple : pour afficher le contenu des tours `tour1`, `tour2` et `tour3` dans la fenêtre graphique, tapez simplement `hanoi.afficher(tour1, tour2, tour3)`



Question 13. Tester la fonction `hanoi.afficher` sur les listes obtenues à la question précédente.

e. Déplacer un disque

Question 14. Ecrire une fonction qui prend en argument deux listes, et qui a pour effet de déplacer le disque situé au sommet de la première tour vers le sommet de la deuxième tour.

Exemple : Si on a `tour2 = [3]` et `tour3 = [2, 1]`, et qu'on exécute l'instruction `deplacer(tour3, tour2)`, on obtient `tour2 = [3, 1]` et `tour3 = [2]`.

Remarque : Un tel déplacement n'est possible que si la tour de départ n'est pas vide, et que la tour d'arrivée ne contient pas un disque plus petit que le disque qu'on s'apprête à déplacer. Si ce n'est pas le cas, utiliser la syntaxe `raise Exception("Texte de l'exception")` pour provoquer une erreur.

f. Résolution du problème

Question 15. Ecrire une fonction prenant les arguments suivants :

- Le nombre de disques à déplacer
- La tour de départ (celle de gauche, qu'on suppose remplie au début)
- La tour intermédiaire (celle du milieu, qu'on suppose vide au début)
- La tour d'arrivée (celle de droite, qu'on suppose vide au début)

Cette fonction doit transférer l'intégralité des disques de la tour de départ à la tour d'arrivée en respectant les règles du jeu (on utilisera donc la fonction déplacer)

Indice 1 : Pensez récursif !

Indice 2 : Essayez de résoudre à la main le problème pour 1 disque, puis pour 2 disques, puis pour 3 disques, etc.

Remarque : Vous pouvez utiliser la fonction `print` pour afficher l'état des tours après chaque déplacement.

★ Question 16. Pour finir, vous pouvez utiliser les indications suivantes pour tester votre fonction de résolution dans la fenêtre graphique :

- Ajouter `import time` au début de votre programme
- Ajouter `time.sleep(0.1)` à la fin de votre fonction `deplacer` (la valeur indiquée entre parenthèses correspond au nombre de secondes d'attente entre chaque déplacement)
- Appeler l'instruction `hanoi.afficher_resolution(f_res, nb_disques)`, avec
  - `f_res` est le nom de votre fonction de résolution
  - `nb_disques` est le nombre de disques à déplacer