TP:Tris et mesure de temps

Partie I

Dans cette partie , on utilise des fonctions écrites dans un fichier (module) qu'on importe.
A partir du site *mathgreen.fr* , télécharger le fichier *listes.py*.

Ce fichier comporte des fonctions qui servent à créer des listes de nombres dans un ordre croissant (cree_liste_croissante(n)), ou dans un ordre décroissant (cree_liste_decroissante(n)), ou dans un ordre quelconque (cree_liste_melangee(n)).
le paramètre n est la longueur de la liste créée.

1. Créer un fichier nommé *mesuretris.py* , puis importer le module *listes.py* à l'aide de l'instruction placée en tête de fichier:

   from listes import *

2. Tester dans le shell les fonctions mentionnées ci-dessus en créant des listes croissantes , décroissantes ou mélangées.

3. Dans un fichier nommé *tris.py* , rassemblez les fonctions de tri étudiées dans le cours et dans la dernière fiche d'exercices , à savoir les tris sélection , insertions et tri à bulles.

   Importez le module *tris.py* dans le fichier *mesuretris.py*.

Partie II

Remarque préliminaire: Dans cette partie on utilisera le tri sélection dans les exemples ou dans les réponses proposées

Le module **timeit** de Python permet de prendre une mesure du temps d'exécution d'une fonction, en secondes. Ce module fournit une fonction *timeit* qui accepte en entrée trois paramètres :

- **setup** permet de préciser à *timeit* les modules à charger pour permettre l'exécution correcte de la fonction (y compris donc le module qui contient la fonction à mesurer),
- **stmt** – pour statement, instruction en anglais – est l'appel de fonction qui sera mesurée (donc avec ses paramètres),
- **number** est le nombre de fois où l'instruction stmt sera exécutée. Le temps mesuré sera le temps cumulé pour toutes ces exécutions.

Remarque: le code Python des deux paramètres setup et stmt sont donnés sous forme d'une **chaîne de caractères**. Par exemple, après avoir importé le module *timeit* :

On peut mesurer le temps d'exécution du tri par sélection sur une liste mélangée de taille 10 :

```
timeit(setup='from tris import tri_selection; from listes import cree_liste_melangee',
stmt='tri_selection(cree_liste_melangee(10))',
number=100)
```

[1] cette activité a été construite à partir d'une activité proposée par Benoit Papegay (université de Lille)

Le résultat obtenu représente le temps total mis pour exécuter 100 fois la fonction *tri_selection* avec l'argument : la liste *cree_liste_melangee(10)*.

1. Testez dans le shell et avec des exemples différents (C'est-à-dire en utilisant des méthodes différentes de création de liste issues du module *listes.py*).
2. Écrire le code d'une fonction nommée ***temps_de_tri(n)*** dont le paramètre n est la longueur d'une liste donnée et qui renvoie la mesure du temps d'exécution pour des listes de cette longueur.(toujours avec le tri *sélection*)
3. Ecrire une variante de la fonction précédente , qu'on nommera ***temps_de_tri_interm(n)*** pour qu'elle accepte maintenant comme paramètre une longueur maximale de liste n, et qui renvoie la liste des temps d'exécution pour des tableaux de tailles 1,2,..,n.

Partie III

1. Écrire une fonction ***courbes_tri***(n) à un paramètre n, qui produit une courbe du temps d'exécution du *tri sélection* pour des listes mélangées dont la taille varie de 1 à la valeur du paramètre donné.
2. (Généralisation à tous les tris)
   (a) Pour étendre le procédé décrit ci-dessus à d'autres tris, on propose d'écrire une fonction à deux paramètres : ***temps_de_tri_qcq***(n,*nom_tri*) qui prend pour paramètres n la longueur maximale d'une liste et *nom_tri* le nom du tri à mesurer.
   (b) Ecrire de même une fonction ***courbes_tri_qcq***(n,*nom_tri*)
   qui produit la courbe des temps d'exécution du tri en question.
3. - Produire une courbe du temps d'exécution du tri sélection, la sauvegarder.
   - Produire une courbe du temps d'exécution du tri insertion, la sauvegarder.
   - Produire une courbe du temps d'exécution du tri à bulles, la sauvegarder.

Annexe

Étant donné un ensemble de valeurs dans une liste *liste*, on peut utiliser le module **pylab** de **matplotlib** pour tracer une courbe où chaque valeur de *liste* à l'indice i est comprise comme le point de coordonnées (i,liste[i]).
Testez le code suivant:

```
import pylab

liste = [1,2,4,8,11,15,19]
pylab.plot(liste)
pylab.show()
```

Plus généralement , on peut tracer la courbe passant par des points dont les coordonnés sont fournies dans deux listes distinctes.Testez le code suivant:

```
x = [1,2,3,5,6,7,10]
y = [1,2,4,8,11,15,16]
pylab.plot(x,y)
pylab.show()
```

On peut améliorer la qualité du graphique produit en spécifiant des titres, et même une grille qui facilite la lecture :

```
NBRE\_ESSAIS = 100
pylab.title('Temps du tri par sélection (pour {:d} essais)'.format(NBRE\_ESSAIS))
pylab.xlabel('taille des listes')
pylab.ylabel('temps en secondes')
pylab.grid()
pylab.show()
```