Informatique PCSI
TP 2 suite : boucles imbriquées

1 Énoncé des exercices

1.1 Exercice 1

Voici la définition d'une fonction qui prend en paramètre une liste de nombres :

```
def remonte(liste):
    for i in range(len(liste)-1):
        if liste[i] > liste[i+1]:
            liste[i], liste[i+1] = liste[i+1], liste[i]
```

On appelle la fonction `remonte` avec une liste de longueur n.
Quelle affirmation est vraie ?

1. Dans la liste modifiée, le plus petit élément est placé au début.
2. Dans la liste modifiée, le plus grand élément est placé à la fin.
3. Le nombre de comparaisons effectuées est exactement égal à n.
4. Le nombre d'échanges effectués est strictement inférieur à $n-1$.

1.2 Exercice 2

On reprend la fonction `remonte` définie dans l'exercice précédent.
On définit une liste `nombres=[12, 5, 13, 8, 11, 6]`. Si on appelle la fonction `remonte` avec en paramètre la liste `nombres`, quel est l'état final de cette liste ?

1.3 Exercice 3

Écrire une fonction `ordre` qui prend en argument une liste de mots et modifie la liste en ordonnant les mots en fonction du nombre de lettres. La fonction ne renvoie rien.
Tester la fonction avec la liste `['toto','bonjour','a','oui','non']`.

Adapter l'algorithme du tri à bulles.

1.4 Exercice 4

L'objectif est de comparer les temps d'exécution du tri à bulles sur deux types de listes : une liste de nombres au hasard et une liste de nombres déjà triée. Utiliser le code du tri à bulles et pour mesurer le temps d'exécution, importer la fonction `time` du module `time`.

1. Construire une liste de 5000 entiers pris au hasard entre 1 et 10000, bornes comprises, puis la liste des 100000 entiers de 0 à 99999, bornes comprises. Mesurer les temps d'exécution du programme du tri à bulles pour trier chacune de ces listes. Quel commentaire peut-on faire sur ces temps ?
2. Comparer avec le temps d'exécution de la méthode `sort` sur une liste de 100000 entiers, choisis de manière aléatoire entre 1 et 100000. La syntaxe est `liste.sort()`.

Pour mesurer le temps d'exécution :

```
from time import time
top = time()
programme à tester
print(time() - top)
```

1.5 Exercice 5

Trier des points

On dispose de points dans le plan muni d'un repère orthonormé d'origine $\mathcal{O}$. Chaque point possède un couple de coordonnées $(x; y)$ représenté par la liste `[x,y]`. Il s'agit de trier ces points en fonction de leur distance à $\mathcal{O}$, de la plus petite à la plus grande.

1. Écrire une fonction `distance2` qui prend en paramètre une liste de deux nombres nommée `point` qui représente un point du plan, (`point` est la liste des coordonnées d'un point P), et renvoie le carré de la distance euclidienne entre ce point et $\mathcal{O}$.
2. Écrire une fonction `compare` qui prend en paramètres deux listes `p1` et `p2` représentant deux points P_1 et P_2 et qui renvoie -1 si P_1 est plus proche de $\mathcal{O}$ que P_2, 1 si P_2 est plus proche de $\mathcal{O}$ que P_1, et 0 si les deux points sont équidistants de $\mathcal{O}$.
3. Écrire une fonction `tri_points` qui prend en paramètre une liste composée de listes de deux nombres représentant des points du plan et qui trie cette liste suivant les distances entre chacun des points et $\mathcal{O}$. Utiliser le tri à bulles.