Informatique PCSI
Corrigé suite TP 2

1 Corrigé des exercices

1.1 Exercice 1

La liste est parcourue du début vers la fin. Donc les différents échanges de valeurs amènent le plus grand élément à la fin. C'est la réponse 2. Exactement $(n-1)$ comparaisons sont effectuées.

1.2 Exercice 2

La liste `nombres` est parcourue. Les valeurs 12 et 5 sont échangées, puis 13 et 8, puis 13 et 11 et enfin 13 et 6. La valeur finale de la liste `nombres` est `[5, 12, 8, 11, 6, 13]`.

1.3 Exercice 3

On adapte le tri à bulles en comparant à chaque étape les longueurs des mots.

```
def ordre(liste):
    for j in range(1, len(liste)):
        trier = True
        for i in range(len(liste)-j):
            # modification du test pour comparer les longueurs des mots
            if len(liste[i]) > len(liste[i+1]):
                liste[i], liste[i+1] = liste[i+1], liste[i]
                trier = False
        if trier:
            break

maliste = ['toto', 'bonjour', 'a', 'oui', 'non']
ordre(maliste)
print(maliste)
```

1.4 Exercice 3

Il faut commencer par importer la fonction `time` avec l'instruction `from time import time` et écrire le code de la fonction du tri à bulles.

1. La fonction `randint` permet de créer la liste aléatoire.

```
from random import randint

liste = [randint(1, 10000) for i in range(5000)]
top = time()
tri_bulles(liste)
print(time() - top)

liste = [i for i in range(100000)]
top = time()
tri_bulles(liste)
```

```
print(time() - top)
```

Le résultat obtenu pour le premier test est de l'ordre de 3 secondes. Pour le second, c'est environ 0,015 seconde. Ces résultats sont bien sûr légèrement variables si on répète l'expérience ou si on change de machine. La différence de coût apparaît nettement, quadratique dans le premier cas et linéaire dans le second.

2.

```
liste = [randint(1, 100000) for i in range(100000)]
top = time()
liste.sort()
print(time() - top)
```

Le temps affiché est environ 0.031 seconde. Il montre l'efficacité de cet algorithme.

1.5 Exercice 5

1.

```
def distance2(point):
    x, y = point
    return x * x + y * y
```

2.

```
def compare(p1, p2):
    d1 = distance2(p1)
    d2 = distance2(p2)
    if d1 < d2:
        return -1
    elif d2 < d1:
        return 1
    else:
        return 0
```

3. Le tri à bulles est adapté. Tester avec la liste `[[3, 5], [2, 1], [0, 7], [-2, 0]]`.

```
def tri_points(liste):
    for j in range(1, len(liste)):
        trier = True
        for i in range(len(liste)-j):
            if compare(liste[i], liste[i+1]) == 1:
                liste[i], liste[i+1] = liste[i+1], liste[i]
                trier = False
        if trier:
            break
```