Exercice 1 :

Ouvrir le fichier **recherche_et_trie.py** et afficher une liste de **NBelts** nombres aléatoirement répartis.

Exercice 2 :

Écrire la fonction **recherche_nombre** qui cherche un nombre dans une liste et renvoie le plus petit indice de ce nombre s'il existe.

Exercice 3 :

Écrire la fonction **recherche_nombre_dicho** qui cherche un nombre dans une liste et renvoie le plus petit indice de ce nombre s'il existe en utilisant une dichotomie. On utilisera la méthode **sort()** pour trier une liste.

Exercice 4 :

Utiliser un compteur global qui s'itère à chaque comparaison dans les deux algorithmes précédents. Quelle est la fonction qui utilise le moins de comparaison.

Est-ce cohérent avec la théorie ?

Exercice 5 :

Écrire la fonction **tri_selection(liste)** qui prend en argument une liste de nombres liste et tri cette liste en utilisant le trie par sélection.

Exercice 6 :

La fonction **tri_fusion** écrite dans le fichier permet de trier une liste en utilisant l'algorithme de tri fusion. Cette fonction itère un compteur **cpt** à chaque comparaison.

Comparer le nombre de comparaisons effectuées entre les algorithmes **tri_selection** et **tri_fusion**.

Est-ce cohérent avec la théorie ?

Exercice 7 :

La fonction **time.clock()** permet de calculer le temps du processeur.

Par exemple en effectuant les commandes :

```
tps1 = time.clock()
L.sort()
tps2 = time.clock()
print tps2-tps1
```

On obtient le temps effectué par le processeur pour trier une liste en utilisant la fonction **L.sort()**.

En utilisant la fonction **time.clock()**, comparer les temps processeur nécessaires pour trier une liste en utilisant les différents algorithmes.