OBJECTIFS : L'objectif de ce Tp est d'étudier La performance de ces algorithmes de tris par sélection ou par insertion.

1. Découverte de la bibliothèque time

```
from time import time,ctime,sleep
```

- La fonction ctime() qui renvoie :
  - la date actuelle formatée si aucun argument donné : ctime() .
    - Qu'obtient-on si on exécute : ctime(0) ? : 'Thu Jan 1 01:00:00 1970'
      L'**heure Unix** ou **heure Posix** est une mesure du temps fondée sur le nombre de secondes écoulées depuis le 1er janvier 1970 00:00:00 UTC, hors secondes intercalaires. Elle est utilisée principalement dans les systèmes qui respectent la norme POSIX[1], dont les systèmes de type Unix, d'où son nom. C'est la représentation POSIX du temps.
  - la date formatée correspondant à l'instant défini par le nombre de seconde écoulée depuis le 1er janvier 1970.
    - Qu'obtient-on si on exécute : ctime(0) ? : la date d'origine de l'heure Posix
    - Qu'obtient-on si on exécute : ctime(2 000 000 000) ? : 'Wed May 18 05:33:20 2033'
      La date POSIX + 2 000 000 000 s

- La fonction sleep() qui permet de réaliser une temporisation.
  - Qu'obtient-on si on exécute : sleep(5) ? : une attente de 5 s

2. Création d'une liste test de nombre aléatoire composée de nombres aléatoires compris entre 0 et 100

```
l  = [randint(0,100) for x in range(10)]
```

3. Importation des fonctions de tris par sélection et par insertion, écrites dans les séances précédentes

```
from time import time,ctime,sleep
from random import randint
from bibliTri import triSelection , triInsertion

l = [randint(0,100) for i in range(10)]
print(l)
depart = time()
l = triSelection(l)
temps = time() - depart
print(l)
print(f"Temps d'exécution : {temps}")
```

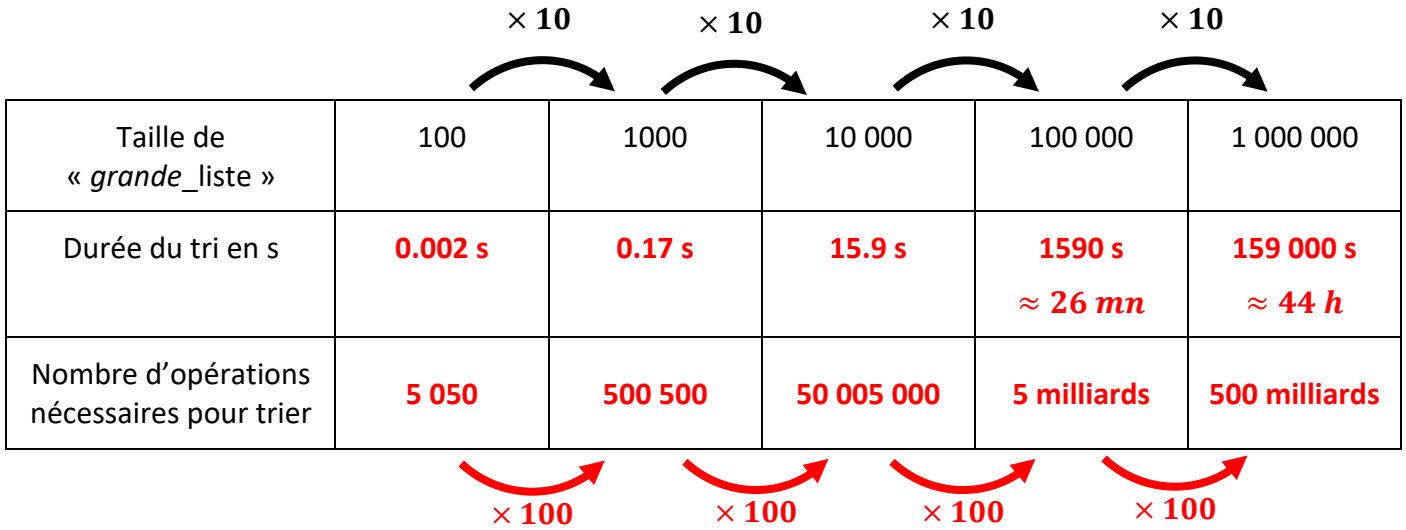
- Exécuter le code pour une liste de taille n = 1000 : 0.016844749450683594

- Exécuter le code pour une liste de taille n = 10 000 : 1.757563591003418

- Exécuter le code pour une liste de taille n = 100 000 : 182.76007843017578

4. Comparaison des temps de calculs

Tri réalisé avec la fonction triSelection() :



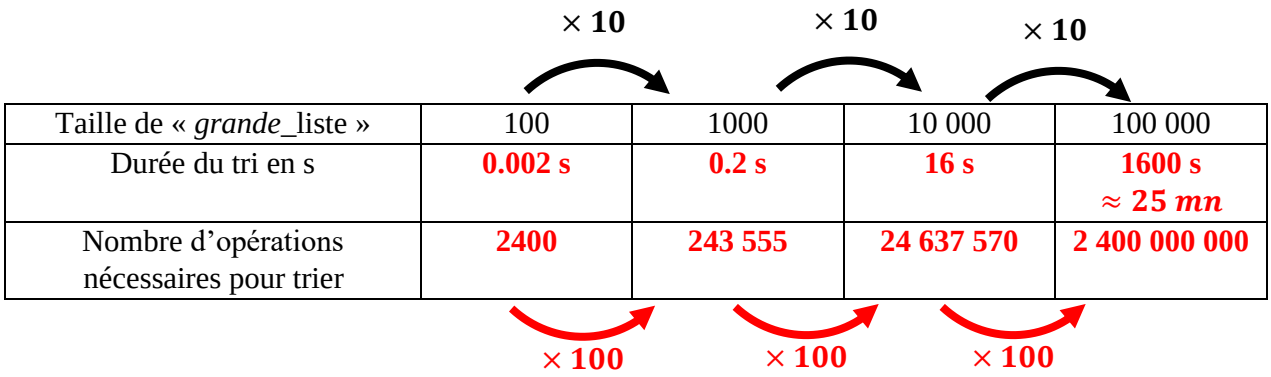
Taille de « *grande*_liste »	100	1000	10 000	100 000	1 000 000
Durée du tri en s	**0.002 s**	**0.17 s**	**15.9 s**	**1590 s** ≈ **26 *mn***	**159 000 s** ≈ **44 *h***
Nombre d'opérations nécessaires pour trier	**5 050**	**500 500**	**50 005 000**	**5 milliards**	**500 milliards**

Complexité calculée par les mathématiques :

$$\frac{n \times (n-1)}{2}$$

Sa complexité est donc O(n^2).

-Tri réalisé avec la fonction triInsertion() :



Taille de « *grande*_liste »	100	1000	10 000	100 000
Durée du tri en s	**0.002 s**	**0.2 s**	**16 s**	**1600 s** ≈ **25 *mn***
Nombre d'opérations nécessaires pour trier	**2400**	**243 555**	**24 637 570**	**2 400 000 000**

ANALYSE : Soit n la taille de la liste à trier et soit t le temps en secondes pour réaliser l'opération de tri. On a : $t \approx k\, n^2$. Le temps de calcul est **proportionnel** au nombre d'opérations réalisées dans les 2 boucles « *for i in range()* ». Comme pour chaque élément de la liste, on **re**parcourt une partie de la liste, le nombre total d'opérations est à peu près égal à $n^2/4$ ici .

On dit que l'algorithme de tri par insertion a une complexité en temps de $O(n^2)$.

Tri réalisé avec la fonction sorted() :

Taille de « *grande*_liste »	100	1 000	10 000	100 000	1 000 000	10 000 000
Durée du tri en s	**0 s**	**0 s**	**0.002 s**	**0.016 s**	**0.22 s**	**2.2 s**

La complexité de la fonction sorted est de : **O(nlogn)**

Le tri réalisé avec la fonction native de Python est bien plus rapide que celui basé sur le tri par sélection. Pourquoi ? …… Le principe de tri utilisé est bien plus efficace que celui du tri par sélection. On verra des exemples similaires au cours de l'année. D'autre part, le script a été optimisé au maximum.

Question : Comment évoluent les temps de calculs en fonction de la taille n de la liste à trier ?

On propose 3 réponses :

- Réponse 1 : Lorsque la taille n est multipliée par 10, les temps de calcul sont multipliés par ≈ 10 aussi -
  Réponse 2 : Lorsque la taille n est multipliée par 10, les temps de calcul sont multipliés par ≈ 100
  Réponse 3 : Autre réponse