```
# ====================================== #
# TRI RAPIDE (QUICK SORT)                #
# Programme de base avec temps de calcul. #
# Version de novembre 2017               #
# ====================================== #

## Importations
from random import randint
from time import perf_counter

## Une première fonction
# ATTENTION ! Cette fonction ne trie pas en place !
# Elle est un codage quasi-direct de l'algorithme... Elle a cet avantage
MAIS
# elle crée beaucoup de listes... Pour limiter la sur-consommation de
mémoire
# qui pourrait en résulter, on vide L pour construire Lg et Ld...
def tri_rapide_rec(L):
    lL = len(L)
    if lL <= 1:
        return L
    else:
        # on choisit ici comme pivot l'élément "central" de la liste
        piv = L.pop(lL//2)
        # Construction des listes Lg et Ld (cf. le cours)
        Lg, Ld = [], []
        # La condition du "while" est satisfaite tant que la liste L n'est
pas
        # vide ...
        while L:
            y = L.pop(0)
            if y < piv:
                Lg.append(y)
            else:
                Ld.append(y)
        return tri_rapide_rec(Lg) + [piv] + tri_rapide_rec(Ld)

## Une deuxième fonction
# Cette fonction trie  en place !

# On a d'abord la fonction de partitionnement de la liste
def partition(L,i_debut,i_fin,i_pivot):
    # 1ère étape : on échange le pivot avec le dernier élément
    L[i_fin], L[i_pivot] = L[i_pivot], L[i_fin]
    # 2ème étape : partitionnement
    j = i_debut
    for i in range(i_debut,i_fin):
        if L[i] < L[i_fin]:
            L[i], L[j] = L[j], L[i]
            j += 1
    # 3ème étape : on replace le pivot au bon endroit
    L[j], L[i_fin] = L[i_fin], L[j]
    # 4ème étape : on renvoie l'indice du pivot
    return(j)

def tri_rapide_rec2(L,i_debut=0,i_fin=None):
    n = len(L)
    # On ne trie que s'il y a vraiment quelque chose à trier ! :)
    if n > 1:
        # Cas du premier appel
```

```
        if i_fin == None:
            i_fin = n - 1
        # Cas général
        if i_debut < i_fin:
            i_pivot = (i_fin + i_debut)//2
            i_pivot = partition(L,i_debut,i_fin,i_pivot)
            tri_rapide_rec2(L,i_debut,i_pivot-1)
            tri_rapide_rec2(L,i_pivot+1,i_fin)

## Pour tester le tri en place...
# Génération d'une liste de nombres entiers aléatoires.
n = int(input('Longueur de la liste à trier ? '))
valmax = int(input('\nValeur maxi des nombres générés ? '))
LNA = [randint(0,valmax) for i in range(n)]
# Affichage de la liste initiale ... si elle n'est pas trop grande ! :)
if n <= 50:
    print('\nListe initiale :')
    print(LNA)

# Tri de la liste LNA des nombres aléatoires + mesure de la durée du tri
t_start = perf_counter()
tri_rapide_rec2(LNA)
t_end = perf_counter()
t_ellapse = t_end - t_start

# Affichage de la liste triée ... si elle n'est pas trop grande ! :)
if n <= 50:
    print('\nListe triée :')
    print(LNA)

# Affichage de la durée du tri
print('\nDurée du tri : ' + str(t_ellapse))

# ================
# FIN DU PROGRAMME
```