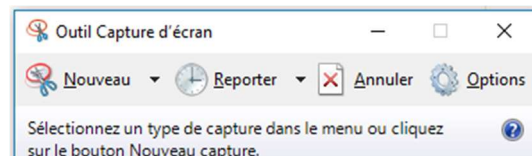


OBJECTIFS : Les objectifs de ce TP sont :

- d'écrire un algorithme qui permette de trier une liste en utilisant un algorithme du type « tri par insertion »,
- d'intégrer cet algorithme dans une fonction,
- de déterminer la durée des différentes opérations réalisées lorsque la liste est de grande taille,
- d'écrire les valeurs des listes triées dans un fichier `.txt`.

DOCUMENT A RENDRE : Ce travail est évalué. Vous en rédigerez un compte-rendu sous le nom « `tp3_nomfamille.doc` » ou « `tp3_nomfamille.odt` » et vous le transférez en fin d'activité par l'intermédiaire de l'onglet transfert du site https://nsibrantly.fr/transferts_terminale.php en utilisant comme identifiant : **tp3** et mot de passe : **tp3**. Ce compte-rendu contiendra :

- les réponses aux différentes questions posées,
- les captures d'écran des morceaux de codes et celles des résultats des exécutions données shell. Pour faire ces captures, utiliser l'*Outil Capture d'écran* de Windows.



écrits dans le

Pour commencer, ouvrir dans Pyzo un fichier nommé `from random import *` « `tri_insertion.py` ». Importer les bibliothèques `random` et `time` : `import time`

1. Création d'une liste de nombres aléatoires

- Créer une liste nommée « `liste_test` » contenant 5 nombres aléatoires compris entre 0 et 10. Afficher le contenu de cette liste.

2. Algorithme de « tri par insertion »

- Découvrez le principe du tri par insertion en utilisant le lien donné dans la rubrique *Documents*.
- Si on applique cette technique à la liste `[7 , 2 , 1 , 4]`, le tri par insertion se fait en 3 étapes :
 - 1^{ère} étape : le 1^{er} élément à placer est 2 : `[7 , 2 , 1 , 4]`. On le compare avec les éléments à gauche et on l'y place correctement : `[2 , 7 , 1 , 4]`
 - 2^{ème} étape : le 2^{ème} élément à placer est 1 : `[2 , 7 , 1 , 4]`. On le compare avec les éléments à gauche et on l'y place correctement : `[1 , 2 , 7 , 4]`
 - 3^{ème} étape : le 3^{ème} élément à placer est 4 : `[1 , 2 , 7 , 4]`. On le compare avec les éléments à gauche et on l'y place correctement : `[1 , 2 , 4 , 7]`

PRINCIPE A RETENIR : On est certain à chaque étape, que les éléments à gauche de l'élément à placer sont triés dans l'ordre.

⇒ Ecrire sur votre compte-rendu les étapes nécessaires pour trier la liste `[9 , 2 , 1 , 7 , 3]`

- Ecrire le code d'une fonction nommée « `tri_insertion(l)` » qui retourne la liste entrée en argument, triée par un algorithme du type « tri par insertion ». Tester cette fonction sur la liste « `liste_test` ».

3. Performance de cet algorithme de tri

- Modifier la fonction « `tri_insertion(l)` » afin qu'elle retourne la liste triée **et** le nombre d'opérations réalisées dans ce tri (insérer un compteur).
- En utilisant la fonction `time()`, déterminer les temps de calcul pour trier les 2 listes. Compléter finalement le tableau ci-dessous :

	Temps mis pour trier (en s)	Nombre d'opérations réalisées pour effectuer ce tri
<code>liste_test</code>		

- Créer une liste nommée « `grande_liste` » par compréhension, qui contient 1000 valeurs de nombres entiers aléatoires compris entre 0 et 100 .
- Utiliser la fonction « `tri_insertion(l)` » pour trier cette liste et écrire les 1000 valeurs triées dans un fichier nommé « `liste_trie.txt` ». Vérifier que le tri a été effectué correctement. Justifier toujours en joignant des copies d'écran.
- En utilisant la fonction `time()`, déterminer le temps de calcul pour trier « `grande_liste` ». Compléter le tableau ci-dessous :

Taille de « <code>grande_liste</code> »	100	1000	10 000	100 000
Durée du tri en s				
Nombre d'opérations nécessaires pour trier				

- Une fonction native de python nommée « `sorted(liste)` » retourne la liste en argument triée. Exécuter dans le shell : `sorted([5,9,1,0])` pour vous en convaincre. Afin d'analyser l'efficacité en temps de calcul, de la fonction « `tri_insertion(l)` », il est intéressant de comparer les durées précédentes avec celles obtenues en employant « `sorted(liste)` ». Compléter le tableau ci-dessous et conclure :

Taille de « <code>grande_liste</code> »	100	1 000	10 000	100 000	1 000 000	10 000 000
Durée du tri en s						