Mini-projet 6 : Autres tris

L'objectif de ce mini-projet est d'utiliser et de mettre en œuvre le **tri bulle** qui est un tri itératif et le **tri rapide** qui est un tri récursif.

Partie 1 : Le tri bulle

Le tri à bulles consiste à parcourir plusieurs fois les éléments consécutifs d'une liste, et à les permuter lorsqu'ils ne sont pas dans l'ordre croissant. On répète ces parcours successifs **tant qu'on a fait au moins une permutation**. Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

Étape 1 : mise en œuvre du tri bulle à la main

Premier exemple : On va détailler chaque étape du tri bulle pour trier la liste [2, 3, 5, 0, 1, 4].

Premier parcours :	[**2, 3**, 5, 0, 1, 4]	(permutation = 0)
	[2, **3, 5**, 0, 1, 4]	(permutation = 0)
	[2, 3, **0, 5**, 1, 4]	(permutation = 1)
	[2, 3, 0, **1, 5**, 4]	(permutation = 2)
	[2, 3, 0, 1, **4, 5**]	(permutation = 3)
Deuxième parcours :	[**2, 3**, 0, 1, 4, 5]	(permutation = 0)
	[2, **0, 3**, 1, 4, 5]	(permutation = 1)
	[2, 0, **1, 3**, 4, 5]	(permutation = 2)
	[2, 0, 1, **3, 4**, 5]	(permutation = 2)
	[2, 0, 1, 3, **4, 5**]	(permutation = 2)
Troisième parcours :	[**0, 2**, 1, 3, 4, 5]	(permutation = 1)
	[0, **1, 2**, 3, 4, 5]	(permutation = 2)
	[0, 1, **2, 3**, 4, 5]	(permutation = 2)
	[0, 1, 2, **3, 4**, 5]	(permutation = 2)
	[0, 1, 2, 3, **4, 5**]	(permutation = 2)
Quatrième parcours :	[**0, 1**, 2, 3, 4, 5]	(permutation = 0)
	[0, **1, 2**, 3, 4, 5]	(permutation = 0)
	[0, 1, **2, 3**, 4, 5]	(permutation = 0)
	[0, 1, 2, **3, 4**, 5]	(permutation = 0)
	[0, 1, 2, 3, **4, 5**]	(permutation = 0)

Deuxième exemple : Détailler chaque étape du tri bulle pour trier la liste [9, 6, 8, 5, 1, 4].

Étape 2 : Programmation du tri bulle en Python

Dans Thonny rédiger une fonction nommée **tri_bulle(liste)** qui prend en argument une liste de type list et qui trie cette liste donnée en paramètre en utilisant le tri bulle. On enregistrera le fichier avec le nom suivant : **mini_projet_6.py** . Rédiger ensuite, après la fonction plusieurs appels de la fonction pour vérifier qu'elle fonctionne correctement.

Étape 3 : Évaluation de la complexité du tri bulle en nombre d'itérations

Rédiger ci-dessous une démonstration, comme pour le tri insertion de la complexité du tri bulle dans le pire des cas, c'est à dire dans le cas de la liste déjà triée dans l'ordre décroissant en **nombre d'itérations**.

Conjecture :

En déduire une conjecture sur la complexité moyenne du tri bulle en nombre d'itérations.

Étape 4 : Vérification de la complexité du tri bulle en nombre d'itérations

Dans le fichier **mini_projet_6.py** rajouter du code à la fin afin de compter le **nombre d'itérations** à l'aide d'un compteur de type **int** dans le pire des cas, c'est à dire la liste déjà triées dans l'ordre décroissant. Exécuter votre programme puis compléter le tableau ci-dessous.

Taille *n*	10	20	50	100	500	1000
Complexité observée						

Partie 2 : Le tri rapide

Le **tri rapide** ou tri pivot (en anglais *quicksort*) est un algorithme de tri fondé sur la méthode de conception "**diviser pour régner**". Il consiste à choisir un pivot (le premier élément par exemple), puis à placer tous les éléments plus petit avant et tous les éléments plus grand après. Puis on fait deux appels récursifs pour trier la partie avant le pivot et celle après le pivot.

Étape 1 : mise en œuvre du tri rapide à la main

Premier exemple : On va détailler chaque étape du tri rapide pour trier la liste

[**5**, 2, 8, 4, 6, 7, 1, 9, 3]

[2, 4, 1, 3, **5**, 8, 6, 7, 9]

[**2**, 4, 1, 3, **5**, **8**, 6, 7, 9]

[1, **2**, 4, 3, **5**, 6, 7, **8**, 9]

[**1**, **2**, **4**, 3, **5**, **6**, 7, **8**, **9**]

[**1**, **2**, 3, **4**, **5**, **6**, 7, **8**, **9**]

[**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**]

[**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**]

Deuxième exemple : Détailler chaque étape du tri rapide pour trier la liste [9, 6, 2, 3, 8, 5, 1, 4, 0, 7].

Étape 2 : Programmation du tri rapide en Python

Dans Thonny, dans le fichier **mini_projet_6.py** rédiger une fonction **récursive** nommée **tri_rapide(liste)** qui prend en argument une liste de type list et qui trie cette liste donnée en paramètre en utilisant le tri rapide avec comme choix de pivot le premier élément de la liste. Rédiger ensuite, après la fonction plusieurs appels de la fonction pour vérifier qu'elle fonctionne correctement.

Étape 3 : Comparaison des différents tris

Réutiliser le TP17 pour tracer les nuages de points des différents tris que l'on a vu :

en bleu : le tri par insertion

en rouge : le tri par sélection

en vert le tri par arbre binaire

en orange le tri fusion

en violet le tri sorted de python

en marron le tri bulle

en noire le tri rapide

Étape 4 * bonus : Tri rapide avec pivot aléatoire

Dans Thonny, dans le fichier **mini_projet_6.py** rédiger une fonction **récursive** nommée **tri_rapide_2(liste)** qui prend en argument une liste de type list et qui trie cette liste donnée en paramètre en utilisant le tri rapide **avec comme choix de pivot un élément pris au hasard dans la liste**. Rédiger ensuite, après la fonction plusieurs appels de la fonction pour vérifier qu'elle fonctionne correctement.

