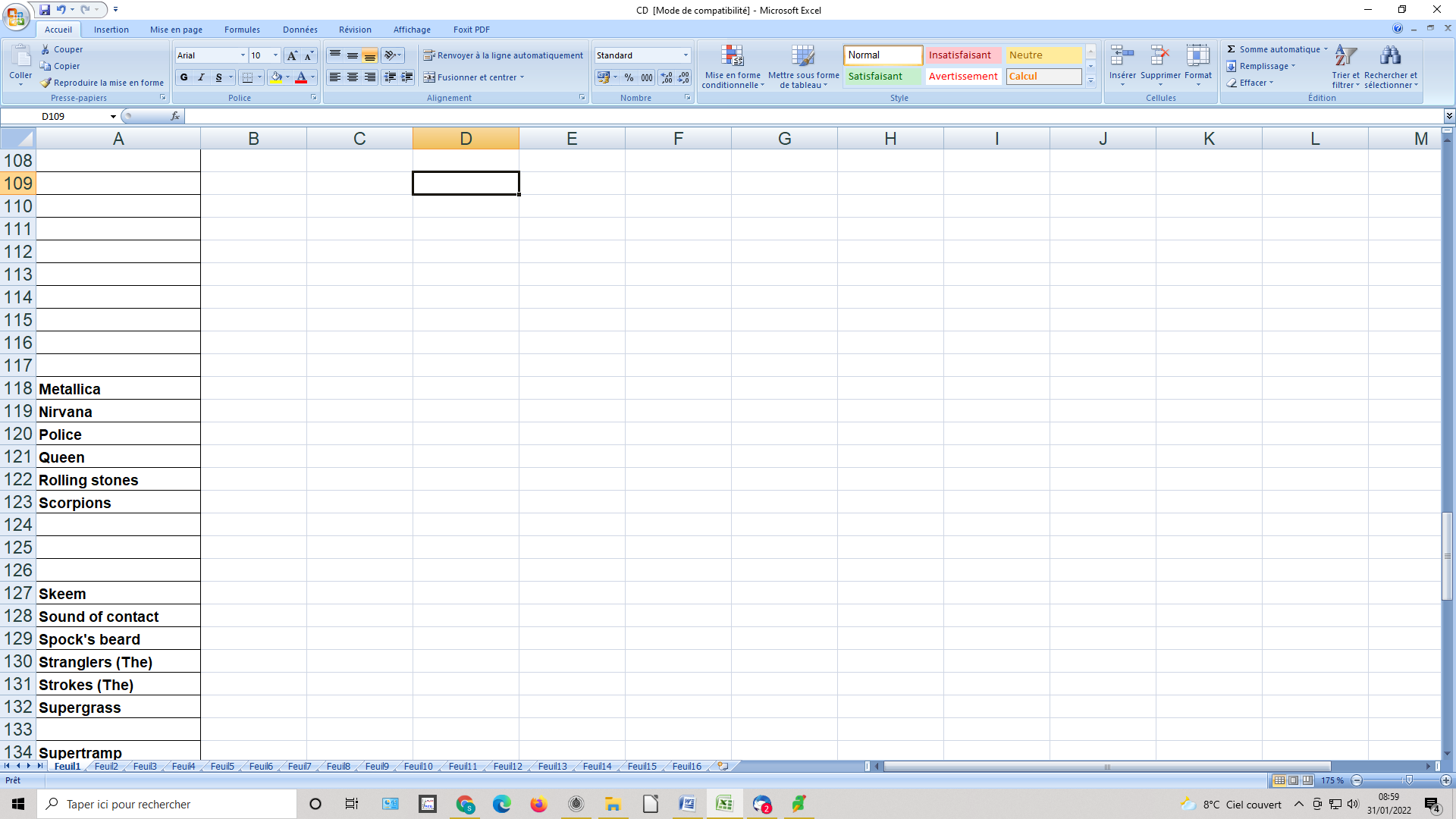
|  |  |  |
| --- | --- | --- |
| Numérique et Sciences Informatiques | | |
| 3h | **Tri de liste** |  |
| **Objectif** : savoir écrire les algorithmes de tri par sélection et de tri par insertion. | | |
| **Matériel**: feuille de papier et Python | | |

**Pourquoi trier des listes ?**

* Pour organiser des données afin d'avoir une meilleure visibilité (exemple : chercher un nom dans une liste triée par ordre alphabétique).
* Pour permettre aux algorithmes de trouver rapidement une information parmi des millions de données (exemple : annuaire en ligne).
* Pour organiser des produits afin de les montrer selon un ordre précis (exemple : prix croissant).
* Pour que les moteurs de recherche puissent proposer les sites les plus populaires en premier.
* Pour utiliser l'intelligence artificielle (exemple : algorithme des k plus proches voisins).

**Méthode de tri par sélection**

**Jeu n°1** : Chaque élève de la classe (sauf un élève maitre) prend un nombre.

Les élèves se mettent en ligne **dans n'importe quel ordre** en montrant leur nombre.

Le maitre va ranger les nombres, dans l'ordre croissant, **dans une nouvelle rangée**.

Déterminez l'**algorithme v1 en pseudo-code** du tri par sélection.

**Utilisez les variables suivantes :**

* L1 : liste composée des nombres à trier
* L2 : nombres de L1 triés dans l'ordre croissants
* i : indice des nombres dans L1
* min : nombre le plus petit de L1

**Code 1\*\*** : Ecrivez la fonction tri\_selection\_v1(L1) qui permet de ranger dans la liste L2 les nombres de la liste L1 dans l'ordre croissant.

Cette fonction pourra modifier la liste L1.

Cette fonction retournera la liste L2.

Vous pouvez utiliser la fonction min().

Vous ne pouvez pas utiliser la méthode sort().

Pour le test : L1=[3,5,4,2,1]

**Jeu n°2** : Même jeu que le jeu n°1 sauf que cette fois le maitre va ranger les nombres, dans l'ordre croissant, **dans la même rangée**.

Le nombre le plus petit de la partie de la liste non triée sera **sélectionné** puis placé au début.

Déterminez l'**algorithme v2 (LE VRAI) en pseudo-code** du tri par sélection.

**Utilisez les variables suivantes :**

* L : liste composée des nombres à trier
* i : indice des nombres dans L
* i\_min : indice du nombre le plus petit de la partie de L non triée

**Code 2\*\*** : Ecrivez la fonction tri\_selection\_v2(L) qui permet de trier les nombres de la liste L dans l'ordre croissant.

Vous pouvez utiliser la fonction min().

Vous pouvez utiliser la méthode index().

Vous ne pouvez pas utiliser la méthode sort().

Cette fonction ne retournera rien car la liste passée en paramètre de la fonction est une référence à cette liste, donc toute modification effectuée sur la liste dans la fonction entraine la modification cette même liste passée en paramètre.

Pour le test : L=[3,5,4,2,1]

Aide : L[2:] ou L[2:len(L)] donne L[4,2,1]

**Code 2b\*\*\* (facultatif)** : Ecrivez la fonction tri\_selection\_v3(L) qui permet de trier les nombres de la liste L dans l'ordre croissant.

Vous pouvez **pas** utiliser la méthode index().

Cette fonction ne retournera rien.

Pour le test : L=[3,5,4,2,1]

**Méthode de tri par insertion**

**Jeu n°3** : Même jeu que le jeu n°2 sauf que cette fois le maitre va ranger les nombres, dans l'ordre croissant, **dans la même rangée** mais avec une méthode différente que celle par sélection.

Le nombre le plus petit de la partie de la liste non triée sera **inséré** au début.

Déterminez l'**algorithme v1 en pseudo-code** du tri par insertion.

**Utilisez les variables suivantes :**

* L : liste composée des nombres à trier
* i : indice des nombres dans L

**Code 3\*\***: Ecrivez la fonction tri\_insertion\_v1(L) qui permet de trier les nombres d'une liste L (quelconque) dans l'ordre croissant.

Cette fonction ne retournera rien.

Vous pouvez utiliser la fonction min().

Vous pouvez utiliser la méthodes insert().

Vous ne pouvez **pas** utiliser la méthode sort().

Pour le test : L=[3,5,4,2,1]

Pour le test (**facultatif**) : créer une grande liste L aléatoirement.

**Jeu n°4** : Même jeu que le jeu n°2 sauf que cette fois le maitre va ranger les nombres, dans l'ordre croissant, **dans la même rangée** mais avec une méthode différente que celle par sélection.

Les nombres seront comparés deux à deux depuis la gauche, le plus petit des deux sera **inséré** à sa place.

Déterminez l'**algorithme v2 (LE VRAI) en pseudo-code** du tri par insertion.

**Utilisez les variables suivantes :**

* L : liste composée des nombres à trier
* i : indice des nombres dans L

**Code 4\*\*\* (facultatif)**: Ecrivez la fonction tri\_insertion\_v2(L) qui permet de trier les nombres d'une liste L (quelconque) dans l'ordre croissant.

Cette fonction ne retournera rien.

Vous ne pouvez **pas** utiliser les méthodes insert() et sort().

Pour le test : L=[3,5,4,2,1]

Pour le test (**facultatif**) : créer une grande liste L aléatoirement.

**Code 5\*** **(facultatif)** : modifiez le programme pour réaliser un tri par **ordre décroissant**.

**Conclusion**

Un algorithme de tri permet d’organiser une liste d’objets selon un ordre déterminé (numérique ou lexicographique le plus souvent).

Il existe plein de méthodes pour trier une liste (par sélection, par insertion, **par fusion\*,** par tas, à bulles ...) mais on va se limiter, cette année, aux tris par sélection et par insertion.

\*Le tri par **fusion** sera vu en terminale NSI.

Les tris utilisant les méthodes par **sélection** ou **insertion** sont quadratique (la durée augmente énormément avec le nombre d'éléments à trier).

Cependant le tri par insertion est plus rapide que celui par sélection si le début de la liste est déjà triée (exemple : liste des habitants d'Angers en 2021 dont on ajoute les nouveaux arrivants de 2022).

Le tri utilisant la méthode **quicksort\*** est logarithmique (la durée augmente peu avec le nombre d'éléments à trier).

\*L'algorithme du **quicksort** est plus complexe que les algorithmes des tris par **sélection** et **insertion**.

Animation tri par sélection : http://lwh.free.fr/pages/algo/tri/tri\_selection.html

Animation tri par insertion : http://lwh.free.fr/pages/algo/tri/tri\_insertion.html

**A tester :**  
L1=[3,5,4,2,1]  
print(L1.sort())

L2=[3,5,4,2,1]

print(sorted(L2))

print(sorted(L2,reverse=True))

L3=['z','a','e','t','b']

print(sorted(L3))

print(sorted(L3,reverse=True))