**1. Comment trier une liste de tuples ?**

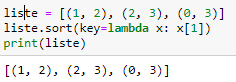
Utilisez la fonction intégrée sorted() de Python ou appelez la méthode sort() sur la liste que vous voulez trier. Ces deux méthodes ont une clé de paramètre optionnelle qui attend une fonction. Passez une fonction ici, qui calcule une valeur clé de chaque tuple de la liste pour trier les tuples selon leurs valeurs clés calculées. Exemple de tri d'une liste de tuples en fonction de leur deuxième entrée

**Syntaxe :**

python

liste = [(1, 2), (2, 3), (0, 3)]  
liste.sort(key=lambda x: x[1])  
print(liste)

**Résultats :**



**2. Comment trier une liste de tuples en fonction d'un élément ou d'une valeur personnalisée ?**

Pour trier les éléments de la liste selon la deuxième entrée des tuples, nous écrivons :

element.sort(key=lambda x: x[1])

ou si vous souhaitez conserver l'ordre initial des objets :

nv\_liste = sorted(element, key=lambda x: x[1])

Notre fonction clé prend un tuple x et renvoie sa deuxième entrée, ainsi, l'ordre final de la liste ne prendra en compte que la deuxième entrée de chaque tuple. Nous pourrions également écrire :

element.sort(key=lambda x: sum(x))

Pour trier les tuples par la somme de leurs entrées. Bien sûr, cela n'est pas applicable à notre exemple, puisque les entrées de nos tuples sont des chaînes de caractères et des entiers.

Enfin, il est également important d'être conscient du comportement par défaut de Python pour le tri des listes de tuples. Cela n'a pas de sens d'écrire :

element.sort(key=lambda x: x[0])

**4. Trier une liste de listes ou de tuples :**

C'est un peu plus compliqué, mais c'est quand même assez facile, alors ne vous inquiétez pas ! La fonction de tri et la fonction de tri prennent toutes deux en compte un argument de mot-clé appelé clé.

Le mot-clé permet de spécifier une fonction qui renvoie ce par quoi vous souhaitez que vos articles soient triés. La fonction reçoit un argument "invisible" qui représente un élément de la liste et renvoie une valeur que vous souhaitez utiliser comme "clé" pour le tri.

Laissez-moi illustrer, pour vos superbes yeux seulement, l'argument du mot-clé !

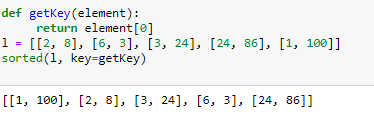
Prenons donc une nouvelle liste et testons-la en la triant par le premier élément de chaque sous-liste

**Syntaxe :**

python

def getKey(element):  
return element[0]  
l = [[2, 8], [6, 3], [3, 24], [24, 86], [1, 100]]  
sorted(l, key=getKey)

**Résultat d’exécution :**



On peut voir ici que la liste est maintenant triée par le premier élément de la sous-liste en ordre croissant. Notez que vous auriez pu utiliser la fonction de tri également, mais personnellement, je préfère la fonction de tri, je l'utiliserai donc dans d'autres exemples.

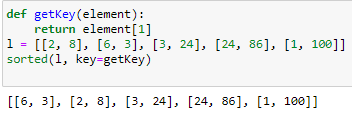
Que s'est-il passé ? Vous vous souvenez de cet argument "invisible" dont je parlais ? C'est ce qui est passé dans la fonction getKey chaque fois que la fonction triée a besoin d'une valeur.

Le tri par le deuxième élément de chaque sous-liste serait aussi simple que de changer la fonction getKey pour celle-ci :

**Syntaxe :**

def getKey(element):  
return element[1]  
l = [[2, 8], [6, 3], [3, 24], [24, 86], [1, 100]]  
sorted(l, key=getKey)

**Résultat d’exécution :**



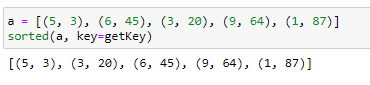
Très bien. Tout va bien. Maintenant, que diriez-vous d'une liste de tuples ?

En fait, c'est exactement la même chose que notre exemple ci-dessus, mais la liste est définie comme telle :

**Syntaxe :**

a = [(5, 3), (6, 45), (3, 20), (9, 64), (1, 87)]  
sorted(a, key=getKey)

**Résultat d’exécution :**



La seule chose qui a changé, c'est que nous recevons maintenant une liste de tuples au lieu d'une liste de listes qui nous est renvoyée.

**5. Trier une liste de tuples par deuxième article :**

**5.1. Utiliser le tri à bulles :**

En utilisant la technique du Tri à Bulles, nous pouvons effectuer le tri. Notez que chaque tuple est un élément de la liste donnée. Accédez au deuxième élément de chaque tuple en utilisant les boucles imbriquées. Cela permet d'effectuer la méthode de tri sur place. La complexité temporelle est similaire à celle du Tri à Bulles, c'est-à-dire O(n^2).

**Syntaxe :**

def Sort\_Tuple(tup):   
lst = len(tup)   
for i in range(0, lst):   
for j in range(0, lst-i-1):   
if (tup[j][1] > tup[j + 1][1]):   
temp = tup[j]   
tup[j]= tup[j + 1]   
tup[j + 1]= temp   
return tup   
tup =[('noveau', 24), ('je', 10), ('membre', 28),   
('bonjour', 5), ('un', 20), ('suis', 15)]   
print(Sort\_Tuple(tup))

**Résultat d’exécution :**

