Algorithme de mises à jour de fichiers Python

Description du projet

Je travaille en tant que professionnel de la sécurité pour une entreprise de santé. mission principale consiste à gérer les accès aux dossiers personnels des patients en fonction des adresses IP des employés. Un fichier, `allow_list.txt`, contient les adresses IP autorisées à accéder à un sous-réseau restreint, tandis qu'un autre fichier, `remove_list.txt`, liste les adresses IP à supprimer. L'objectif de ce projet est de créer un algorithme en Python qui met à jour automatiquement la liste d'autorisations en supprimant les adresses IP identifiées dans la liste de suppression.

Ouvrez le fichier contenant la liste d'autorisations

Je commence par affecter le nom du ficher “allow_list.txt” à la variable “import_file”.

```
# Affecter le nom du fichier à la variable `import_file`

import_file = "data/allow_list.txt"
```

La première étape de l'algorithme consiste à ouvrir le fichier `allow_list.txt` contenant les adresses IP autorisées. Pour cela, j'utilise l'instruction `with` combinée à la fonction `open()`.

```
# Créer une instruction `with` pour lire le contenu initial du fichier

with open(import_file, "r") as file:
```

- **Instruction `with` :** Je l'utilise pour garantir une gestion sécurisée du fichier, en m'assurant qu'il est bien fermé une fois les opérations terminées.
- **Fonction `open()` :** J'ouvre le fichier `allow_list.txt` (premier argument) en mode lecture 'r' second argument pour accéder à son contenu.
- **Mot-clé `as` :** Il permet de renommer le fichier ouvert en une variable, ici `file`, pour l'utiliser facilement dans le bloc d'instructions suivant.
- **Variable `file` :** Représente le fichier ouvert, ce qui me permet de lire, écrire ou effectuer d'autres opérations sur ce fichier en utilisant cette variable.

Lisez le contenu du fichier

Pour pouvoir lire le contenu du fichier ”allow_list.txt”, j’utilise la fonction “.read()” pour le convertir en chaîne.

```
# Créer une instruction `with` pour lire le contenu initial du fichier
with open(import_file, "r") as file:
  # Utiliser `.read()` pour lire le fichier importé et le stocker dans une variable nommée `ip_addresses`
    ip_addresses = file.read()
```

- J'ai utilisé l'argument `'r'` (pour lecture) avec la fonction `open()` dans le corps de l'instruction `with`.
- Cela me permet d'utiliser la méthode `.read()` pour lire le contenu du fichier.
- La méthode `.read()` convertit tout le contenu du fichier en une chaîne de caractères.
- J'ai appliqué cette méthode sur la variable `file`, définie dans le bloc `with`.
- Le résultat de cette opération est ensuite stocké dans la variable `ip_addresses`, ce qui me permet de manipuler cette chaîne de caractères par la suite.

Convertissez la chaîne en liste

Pour pouvoir supprimer des adresses IP individuelles de la liste d'autorisations, je dois convertir la chaîne de caractères `ip_addresses` en une liste.

```
# Utiliser `.split()` pour convertir `ip_addresses` d'une chaîne à une liste
ip_addresses = ip_addresses.split()
```

- Pour cela, j'ai utilisé la méthode `.split()`, qui divise la chaîne en éléments individuels (chaque élément étant une adresse IP) et les stocke dans une liste.

Itération dans la liste de suppression

Pour savoir si des IP adresses qui font parties de la liste “remove_list” sont présent dans la liste “ip_addresses”, j’utilise l’instruction itérative “for”.

```
# Créer une instruction itérative
# Nommer la variable de boucle `element`
# Créer une boucle pour parcourir `remove_list`

for element in remove_list:
```

- **Boucle `for` :** J'utilise une boucle pour parcourir chaque adresse IP de la liste `remove_list`.
- **Condition `if` :** Je vérifie si l'adresse IP actuelle figure dans la liste `ip_addresses` que j'ai précédemment créée.

Supprimez les adresses IP figurant sur la liste de suppression

Je dois supprimer toute adresse IP de la liste `ip_addresses` qui se trouve aussi dans la liste `remove_list`.

```
  # Utiliser la méthode `.remove()` pour supprimer
  # les éléments de `ip_addresses`

    ip_addresses.remove(element)
```

- La méthode `.remove()` est utilisée pour retirer un élément spécifique d'une liste.
- Dans ma boucle `for`, chaque adresse IP de `remove_list` est vérifiée.
- Si l'adresse IP est trouvée dans `ip_addresses`, je l'enlève en utilisant `.remove()`, en passant la variable de boucle `element` comme argument.

Mettez à jour le fichier avec la liste révisée d'adresses IP

Pour mettre à jour le fichier `allow_list.txt`, je dois d'abord convertir la liste `ip_addresses` en une chaîne de caractères.

```
# Convertir `ip_addresses` en une chaîne afin de pouvoir l'écrire dans un fichier texte

ip_addresses = " ".join(ip_addresses)
```

- J'utilise la méthode `.join()`, qui permet de fusionner tous les éléments d'une liste en une seule chaîne.
- Dans ce cas, j'ai utilisé un espace `" "` pour séparer chaque adresse IP dans la chaîne.

Ensuite, j'ai utilisé la méthode `.write()` pour écrire cette chaîne dans le fichier `allow_list.txt`, remplaçant l'ancien contenu par la nouvelle liste d'adresses IP.

```
# Créer l'instruction `with` pour réécrire le fichier original

with open(import_file, "w") as file:

  # Réécrire le fichier en remplaçant son contenu par `ip_addresses`

    file.write(ip_addresses)
```

- J'utilise l'instruction `with open(import_file, "w") as file` pour ouvrir le fichier en mode écriture (`"w"`). Cela permet de modifier le fichier.
- L'argument `"w"` signifie que je vais écrire dans le fichier et que tout le contenu existant sera remplacé.
- Avec la méthode `.write(ip_addresses)`, j'écris la nouvelle chaîne d'adresses IP dans le fichier, écrasant ainsi l'ancien contenu.

Gestion des Commentaires et de l'Indentation

Pour faciliter la maintenance future du code, j'ai inclus des commentaires à chaque étape clé du script en utilisant `#`. Ces commentaires m'aident à comprendre rapidement ce que fait chaque partie du code, ce qui est crucial si je dois le modifier plus tard. De plus, j'ai veillé à respecter l'indentation correcte, ce qui est essentiel en Python pour s'assurer que le code s'exécute correctement.

Synthèse

J'ai développé un algorithme automatisé pour gérer les droits d'accès réseau dans une organisation. L'utilisation de la fonction `with` et de la méthode `open()` garantit une gestion sûre des fichiers, tandis que les méthodes `.read()`, `.split()`, et `.remove()` permettent un traitement efficace des données. La boucle `for` me permet de parcourir systématiquement les adresses IP à supprimer, et la méthode `.write()` me permet d'enregistrer les modifications dans le fichier.