ACTUALIZAR UN ARCHIVO A TRAVÉS DE UN ALGORITMO DE PYTHON

En mi organización, controlamos el acceso a contenido restringido mediante una lista de direcciones IP autorizadas, almacenadas en el archivo "allow_list.txt". También manejamos una lista separada de direcciones IP que ya no deberían tener acceso. Para simplificar este proceso, diseñé un algoritmo que actualiza automáticamente el archivo "allow_list.txt" eliminando las direcciones IP desautorizadas.

Proceso de Actualización del Archivo de Permisos

1. **Apertura y Lectura del Archivo**: Para comenzar, abrí el archivo "allow_list.txt" en modo lectura. Usando una instrucción with, que facilita la gestión de recursos, especifiqué el archivo a través de la variable import_file y lo abrí en modo lectura ("r"). Así, pude acceder a las direcciones IP autorizadas sin necesidad de cerrar manualmente el archivo.

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"
```

```
# Build `with` statement to read in the initial contents of the file

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

2. **Lectura del Contenido**: Una vez abierto, leí el contenido del archivo en una cadena utilizando el método .read(). Esto convierte los datos del archivo en una cadena, lo que me permite manipularlos dentro del algoritmo. Almacené la cadena resultante en la variable ip_addresses.

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()
```

3. **Conversión a Lista**: Para facilitar la eliminación de direcciones IP, transformé la cadena en una lista aplicando el método .split(), que convierte ip_addresses en una lista de direcciones IP. Esta lista es más fácil de manejar para eliminar entradas individuales.

```
# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()
```

4. **Iteración y Eliminación**: Con la lista remove_list (direcciones IP a eliminar), ejecuté un bucle for que recorre cada dirección IP de esta lista. Para cada dirección, verifiqué si estaba presente en ip_addresses. Si lo estaba, usé el método .remove() para eliminarla, evitando errores en caso de que no se encuentre.

```
# Build iterative statement
# Name loop variable `element`
# Loop through `remove_list`

for element in remove_list:
```

```
for element in remove_list:

  # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

      # use the `.remove()` method to remove
      # elements from `ip_addresses`

        ip_addresses.remove(element)
```

5. **Actualización del Archivo**: Finalmente, volví a convertir ip_addresses a una cadena utilizando .join(), con cada dirección IP en una nueva línea. Esto permite escribir la lista revisada en el archivo "allow_list.txt". Usé el método .write() en un nuevo bloque with, abriendo el archivo en modo escritura ("w"), para reemplazar el contenido existente con la lista actualizada.

```
# Convert `ip_addresses` back to a string so that it can be written into the text file

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

```
  # Build `with` statement to rewrite the original file

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

    # Rewrite the file, replacing its contents with `ip_addresses`

    file.write(ip_addresses)
```

Resumen del Algoritmo

El algoritmo simplifica el control de acceso al contenido restringido al automatizar la actualización de direcciones IP en el archivo "allow_list.txt". Lee y convierte el archivo en una lista manipulable, elimina las IP de remove_list, y luego reescribe el archivo con la lista de IPs revisada. Esto asegura que las direcciones IP no autorizadas ya no puedan acceder al contenido restringido.