Actualizar un archivo mediante un algoritmo en Python

Descripción del proyecto

En mi organización, el acceso a contenido restringido está controlado mediante una lista de IPs permitidas (*allow list*). El archivo **"allow_list.txt"** identifica estas direcciones IP. Una lista de exclusión separada (*remove list*) identifica las direcciones IP que ya no deberían tener acceso a ese contenido. Creé un algoritmo para automatizar la actualización del archivo **"allow_list.txt"** y eliminar las direcciones IP que deben ser removidas.

Abrir el archivo que contiene la lista permitida

Para la primera parte del algoritmo, abrí el archivo **"allow_list.txt"**. Primero, asigné este nombre de archivo como una cadena a la variable `import_file`:

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

import_file = "allow_list.txt"
```

Luego, utilicé una sentencia `with` para abrir el archivo:

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

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

En mi algoritmo, la sentencia `with` se utiliza con la función `.open()` en modo lectura ("r") para abrir el archivo de la lista permitida con el fin de leerlo. El objetivo de abrir el archivo es acceder a las direcciones IP almacenadas. La palabra clave `with` ayuda a gestionar los recursos cerrando el archivo automáticamente al finalizar el bloque. En el código `with open(import_file, "r") as file:`, la función `open()` tiene dos parámetros: el primero identifica el archivo a importar y el segundo indica lo que quiero hacer con él. En este caso, "r" indica que quiero leerlo. El código también utiliza la palabra clave `as` para asignar el resultado de la función `open()` a la variable `file`, mientras se trabaja dentro del bloque `with`.

Leer el contenido del archivo

Para leer el contenido del archivo, utilicé el método `.read()` para convertirlo en una cadena de texto:

```
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()
```

Cuando se usa una función `open()` con el argumento "r" para lectura, se puede llamar a `.read()` dentro del cuerpo de la sentencia `with`. El método `.read()` convierte el archivo en una cadena, lo cual permite leerlo. Apliqué el método `.read()` a la variable `file` identificada en el bloque `with`. Luego, asigné la salida de tipo cadena de este método a la variable `ip_addresses`.

En resumen, este código lee el contenido del archivo **"allow_list.txt"** en formato de texto, lo que me permite organizar y extraer los datos posteriormente en mi programa en Python.

Convertir la cadena en una lista

Para poder eliminar direcciones IP individuales de la lista, necesitaba convertirla en una lista. Por eso, usé el método `.split()` para convertir la cadena `ip_addresses` en una lista:

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

ip_addresses = ip_addresses.split()
```

La función `.split()` se llama al final de una variable de tipo cadena. Convierte el contenido de una cadena en una lista. El objetivo de dividir `ip_addresses` en una lista es facilitar la eliminación de direcciones IP. Por defecto, la función `.split()` divide el texto por espacios en blanco. En este algoritmo, toma los datos de `ip_addresses` (una cadena con direcciones IP

separadas por espacios) y los convierte en una lista de direcciones IP. Para almacenar esta lista, la reasigné a la misma variable `ip_addresses`.

Iterar a través de la lista de exclusión

Una parte clave del algoritmo consiste en iterar a través de las direcciones IP que están en `remove_list`. Para esto, incorporé un bucle `for`:

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

for element in remove_list:
```

El bucle `for` en Python repite un bloque de código para una secuencia determinada. El propósito de este bucle es aplicar instrucciones específicas a cada elemento de la secuencia. La palabra clave `for` inicia el bucle, seguida por la variable de iteración `element` y la palabra clave `in`. Esta última indica que se debe recorrer la secuencia `ip_addresses`, asignando cada valor a `element`.

Eliminar las direcciones IP que estén en la lista de exclusión

Mi algoritmo requiere eliminar cualquier dirección IP de `ip_addresses` que también esté en `remove_list`. Como no había duplicados, pude usar el siguiente código:

```
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)
```

Primero, dentro del bucle `for`, creé una condición para evaluar si `element` se encontraba en `ip_addresses`. Hice esto porque aplicar `.remove()` a un elemento no presente genera un error.

Luego, dentro de esa condición, apliqué `.remove()` a `ip_addresses`, pasando como argumento la variable `element`, para que cada dirección IP en `remove_list` fuera eliminada de `ip_addresses`.

Actualizar el archivo con la lista revisada de direcciones IP

Como paso final, necesitaba actualizar el archivo **"allow_list.txt"** con la lista revisada. Para ello, primero convertí la lista nuevamente en una cadena utilizando `.join()`:

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

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

El método `.join()` combina todos los elementos de un iterable en una cadena. Se aplica sobre una cadena que indica cómo se deben separar los elementos al unirlos. En este caso, usé `"\n".join()` para que cada dirección IP aparezca en una nueva línea al convertir la lista `ip_addresses` en una cadena.

Después, utilicé otra sentencia `with` y el método `.write()` para actualizar el archivo:

```
# 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)
```

Esta vez, usé el argumento `"w"` con la función `open()` dentro del bloque `with`. Este argumento indica que quiero escribir en el archivo, reemplazando su contenido. El método `.write()` escribe una cadena en un archivo especificado y reemplaza cualquier contenido existente.

En este caso, quería escribir la nueva lista de direcciones IP en el archivo **"allow_list.txt"**. De esta manera, las direcciones IP eliminadas ya no tendrán acceso al contenido restringido. Para reescribir el archivo, apliqué `.write()` al objeto `file` y pasé la variable `ip_addresses` como argumento.

Resumen

Creé un algoritmo que elimina direcciones IP contenidas en la variable `remove_list` del archivo **"allow_list.txt"** de direcciones IP permitidas. Este algoritmo abre el archivo, lo convierte en una cadena para leerlo, y luego en una lista almacenada en `ip_addresses`. Luego recorre `remove_list`, y si algún elemento se encuentra en `ip_addresses`, lo elimina con `.remove()`. Después de eso, usa `.join()` para volver a convertir `ip_addresses` en una cadena y sobrescribe el archivo **"allow_list.txt"** con la lista actualizada.