Algorithm for file updates in Python

Project description

En mi organización, el acceso a contenido restringido se controla mediante una lista de direcciones IP permitidas (allow list).El archivo **"allow_list.txt"** identifica estas direcciones IP. Una lista separada de direcciones a eliminar (remove list) identifica las IP que ya no deberían tener acceso a este contenido.

Creé un algoritmo para automatizar la actualización del archivo **"allow_list.txt"** y eliminar estas direcciones IP que ya no deben tener acceso.

Open the file that contains the allow list

Para la primera parte del algoritmo, abrí el archivo **"allow_list.txt"**. Primero asigné el nombre de este 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 instrucción **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 instrucción **with** se utiliza junto con la función **.open()** en modo de lectura para abrir el archivo de la lista permitida. El propósito de abrir el archivo es permitirme acceder a las direcciones IP almacenadas en él. La palabra clave **with** ayuda a gestionar los recursos, ya que cierra el archivo automáticamente al salir del bloque **with**.

En el código `with open(import_file, "r") as file:`, la función **open()** tiene dos parámetros:

- El primero identifica el archivo que quiero importar.
- El segundo indica lo que quiero hacer con el archivo; en este caso, `"r"` indica que quiero leerlo.

El código también utiliza la palabra clave **as** para asignar una variable llamada **file**; esta variable almacena el resultado de la función **.open()** mientras trabajo dentro del bloque **with**.

Read the file contents

Para poder 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 utiliza la función **.open()** con el argumento `"r"` para “read” (leer), puedo llamar al método **.read()** dentro del cuerpo de la instrucción **with**. El método **.read()** convierte el contenido del archivo en una cadena de texto y me permite leerlo. Apliqué el método **.read()** a la variable **file** definida en la instrucción **with**. Luego asigné la cadena resultante de este método a la variable **ip_addresses**.

En resumen, este código lee el contenido del archivo **"allow_list.txt"** y lo convierte en una cadena, lo que me permite usar esa cadena más adelante para organizar y extraer datos dentro de mi programa en Python.

Convert the string into a list

Para poder eliminar direcciones IP individuales de la lista permitida, necesitaba que estuviera en formato de lista. Por lo tanto, a continuación utilicé 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 agregándola al final de una variable de tipo cadena. Funciona convirtiendo el contenido de una cadena en una lista. El propósito de dividir **ip_addresses** en una lista es facilitar la eliminación de direcciones IP de la lista permitida. De forma

predeterminada, la función **.split()** divide el texto por espacios en blanco y crea elementos individuales en la lista. En este algoritmo, la función **.split()** toma los datos almacenados en la variable **ip_addresses**, que es una cadena de direcciones IP separadas por espacios, y convierte esa cadena en una lista de direcciones IP. Para almacenar esta lista, reasigné el resultado nuevamente a la variable **ip_addresses**.

Iterate through the remove list

Una parte clave de mi algoritmo consiste en iterar a través de las direcciones IP que son elementos de **remove_list**. Para hacerlo, 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 código para una secuencia específica. El propósito general del bucle **for** en un algoritmo como este es aplicar ciertas instrucciones a todos los elementos de una secuencia. La palabra clave **for** inicia el bucle. Le sigue la variable del bucle (**element**) y luego la palabra clave **in**. La palabra clave **in** indica que se debe iterar a través de la secuencia **ip_addresses** y asignar cada valor a la variable del bucle **element**.

Remove IP addresses that are on the remove list

Mi algoritmo requiere eliminar cualquier dirección IP de la lista permitida (**ip_addresses**) que también esté contenida en **remove_list**. Como no había direcciones duplicadas en **ip_addresses**, pude usar el siguiente código para hacerlo:

```
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 de mi bucle **for**, creé una condición que evaluaba si la variable del bucle (**element**) se encontraba o no en la lista **ip_addresses**. Hice esto porque aplicar **.remove()** a elementos que no se encuentran en **ip_addresses** produciría un error.
Luego, dentro de esa condición, apliqué **.remove()** a **ip_addresses**. Pasé la variable del bucle **element** como argumento para que cada dirección IP que estuviera en **remove_list** fuera eliminada de **ip_addresses**.

Update the file with the revised list of IP addresses

As a final step in my algorithm, I needed to update the allow list file with the revised list of IP addresses. To do so, I first needed to convert the list back into a string. I used the `.join()` method for this:

```
# 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 sola cadena. El método **.join()** se aplica a una cadena que contiene los caracteres que separarán los elementos del iterable una vez unidos en una cadena. En este algoritmo, utilicé el método **.join()** para crear una cadena a partir de la lista **ip_addresses**, de modo que pudiera pasarla como argumento al método **.write()** al escribir en el archivo **"allow_list.txt"**. Usé la cadena **("\n")** como separador para indicarle a Python que coloque cada elemento en una nueva línea.

Luego, utilicé otra instrucción **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 utilicé un segundo argumento, `"w"`, con la función **open()** en mi instrucción **with**. Este argumento indica que quiero abrir un archivo para sobrescribir su contenido.
Cuando se usa el argumento `"w"`, puedo llamar al método **.write()** dentro del cuerpo de la instrucción **with**. El método **.write()** escribe datos de tipo cadena en un archivo específico y reemplaza cualquier contenido existente.

En este caso, quería escribir la lista actualizada de direcciones permitidas como una cadena en el archivo **"allow_list.txt"**. De esta manera, el contenido restringido ya no será accesible para ninguna de las direcciones IP que fueron eliminadas de la lista permitida.
Para reescribir el archivo, agregué el método **.write()** al objeto de archivo **file**, que identifiqué en la instrucción **with**. Pasé la variable **ip_addresses** como argumento para indicar que el contenido del archivo especificado en la instrucción **with** debía ser reemplazado por los datos almacenados en esta variable.

Summary

Creé un algoritmo que elimina del archivo **"allow_list.txt"** las direcciones IP identificadas en la variable **remove_list**. Este algoritmo implicó abrir el archivo, convertirlo en una cadena para poder leerlo y luego convertir esa cadena en una lista almacenada en la variable **ip_addresses**. Después iteré a través de las direcciones IP en **remove_list**. En cada iteración, evalué si el elemento formaba parte de la lista **ip_addresses**. Si lo era, apliqué el método **.remove()** para eliminar ese elemento de **ip_addresses**. Después de esto, utilicé el método **.join()** para convertir nuevamente **ip_addresses** en una cadena, de modo que pudiera sobrescribir el contenido del archivo **"allow_list.txt"** con la lista revisada de direcciones IP.