Actualizar un archivo a través de un algoritmo de Python

Descripción del proyecto

Como profesional de la seguridad en una empresa del sector de atención médica, donde la protección de los registros personales de los pacientes es una prioridad. Una de mis responsabilidades clave es gestionar el acceso al contenido restringido, el cual se regula mediante una lista de direcciones IP autorizadas que identifica al personal con acceso permitido.

Este acceso está controlado por un archivo llamado `"allow_list.txt"`, que contiene las direcciones IP de los empleados autorizados para interactuar con registros sensibles. Dado que los cambios en el personal y las funciones ocurren con regularidad, también existe una lista de eliminación que identifica las direcciones IP asociadas a empleados que ya no deberían tener acceso.

Para garantizar que estas actualizaciones se realicen de manera eficiente y sin errores, he diseñado un algoritmo en Python que automatiza este proceso. El algoritmo verifica si la lista de autorizaciones contiene direcciones IP presentes en la lista de eliminación. En caso afirmativo, elimina esas direcciones IP del archivo `"allow_list.txt"`, asegurando así que solo el personal autorizado pueda acceder a la subred restringida.

Abrir el archivo con la lista de permisos

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

```
# Asigna a `import_file` el nombre del archivo

import_file = "allow_list.txt"
```

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

```
# Crea la sentencia `with` para leer el contenido inicial del archivo

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

En mi algoritmo, la sentencia `with` se usa con la función `.open()` en modo de lectura para abrir el archivo de lista de permitidos con el fin de leerlo. El propósito de abrir el archivo es permitirme acceder a las direcciones IP almacenadas en el archivo de la lista de permitidos. La palabra clave `with` ayudará a administrar los recursos al cerrar el archivo después de salir de la sentencia `with`. 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, lo que quiero hacer con el archivo. En este caso, `"r"` indica que quiero leerlo. El código también usa la palabra clave `as` para asignar una variable llamada `file`; `file` almacena la salida de la función `.open()` mientras trabajo dentro de la sentencia `with`.

Leer el contenido del archivo

Para leer el contenido del archivo, utilicé el método `.read()` para convertirlo en la cadena.

```
# Crea la sentencia `with` para leer el contenido inicial del archivo
with open(import_file, "r") as file:
  # Usa `.read()` para leer el archivo importado y almacenarlo en una variable llamada `ip_addresses`
  ip_addresses = file.read()
```

Al usar una función `.open()` que incluye el argumento `"r"` para "read", puedo llamar a la función `.read()` en el cuerpo de la sentencia `with`. El método `.read()` convierte el archivo en una cadena y me permite leerlo. Apliqué el método `.read()` a la variable `file` identificada en la sentencia `with`. Luego, asigné la salida de cadena de este método a la variable `ip_addresses`.

En resumen, este código lee el contenido del archivo `"allow_list.txt"` en un formato de cadena que me permite usar más tarde la cadena para organizar y extraer datos en mi programa Python.

Convertir la cadena en una lista

Para eliminar direcciones IP individuales de la lista de permisos, necesitaba que estuviera en formato de lista. Por lo tanto, utilicé el método `.split()` para convertir la cadena `ip_addresses` en una lista:

```
# Usa `.split()` para convertir `ip_addresses` de una cadena a una lista
ip_addresses = ip_addresses.split()
```

Se llama a la función `.split()` al agregarla a una variable de cadena. Lo que hace es convertir 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 de permitidos. De forma predeterminada, la función `.split()` separa el texto por espacios en blanco en elementos de lista. En este algoritmo, la función `.split()` toma los datos almacenados en la variable `ip_addresses` (que es una cadena de direcciones IP que están separadas por un espacio en blanco) y convierte esta cadena en una lista de direcciones IP. Para almacenar esta lista, la reasigné a la variable `ip_addresses`.

Iterar a través de la lista de eliminación

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

```
# Crea una sentencia iterativa
# Nombra la variable de bucle `element`
# Recorre en bucle `remove_list`

for element in remove_list:
```

El bucle `for` en Python repite el código para una secuencia especificada. El propósito general del bucle `for` en un algoritmo de Python como este es aplicar sentencias de código específicas a todos los elementos de una secuencia. La palabra clave `for` inicia el bucle `for`. Le sigue la variable del bucle `element` y la palabra clave `in`. La palabra clave `in` indica iterar a través de la secuencia `ip_addresses` y asignar cada valor a la variable de bucle `element`.

Eliminar direcciones IP que están en la lista de eliminación

Mi algoritmo requiere eliminar las direcciones IP de la lista de permitidos `ip_addresses`, que también estén en `remove_list`. Puesto que todos los elementos de `remove_list` también están en la lista `ip_addresses` y que la lista `ip_addresses` no contiene duplicados, pude incorporar el método `.remove()` en el cuerpo de mi bucle `for` de la siguiente manera:

```
for element in remove_list:

  # usa el método `.remove()` para eliminar
  # elementos de `ip_addresses`

    ip_addresses.remove(element)
```

Debido a que las direcciones IP en `remove_list` se deben eliminar de la lista `ip_addresses`, apliqué `.remove()` a `ip_addresses`. Pasé la variable de bucle `element` como argumento para que cada dirección IP que estaba en `remove_list` se eliminara de `ip_addresses`.

Actualizar el archivo con la lista revisada de direcciones IP

Como paso final en mi algoritmo, necesitaba actualizar el archivo de la lista de permitidos con la lista revisada de direcciones IP. Para ello, primero tuve que convertir la lista de nuevo en una cadena. Para hacerlo, usé el método `.join()`:

```
# Vuelve a convertir `ip_addresses` en una cadena para que se pueda escribir en un archivo de texto

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

El método `.join()` combina todos los elementos de un iterable en una cadena. Se aplica a una cadena que contiene caracteres que separarán los elementos en el 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` para poder pasarla como argumento al método `.write()` al escribir en el archivo `"allow_list.txt"`. Utilicé un solo espacio `(" ")` como separador.

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

```
# Crea una sentencia `with` para reescribir el archivo original

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

  # Para reescribir el archivo, reemplaza su contenido con `ip_addresses`

  file.write(ip_addresses)
```

Esta vez, utilicé un segundo argumento `"w"` con la función `open()` en mi sentencia `with`. Este argumento indica que quiero abrir un archivo para escribir en su contenido. Al usar este argumento `"w"`, puedo llamar a la función `.write()` en el cuerpo de la sentencia `with`. La función `.write()` escribe los datos de la cadena en un archivo especificado y reemplaza el contenido de archivo existente.

En este caso, quería escribir la lista de permisos actualizada como una cadena en el archivo `"allow_list.txt"`. De esta manera, las direcciones IP que se hayan eliminado de la lista de permitidos ya no podrán acceder al contenido restringido. Para reescribir el archivo, agregué la función `.write()` al objeto de archivo `file` que identifiqué en la sentencia `with`. Pasé la variable `ip_addresses` como argumento para especificar que el contenido del archivo especificado en la sentencia `with` se reemplace con los datos de esta variable.

Resumen

Creé un algoritmo que elimina las direcciones IP identificadas en una variable `remove_list` del archivo `"allow_list.txt"` con las direcciones IP aprobadas. Este algoritmo implicaba abrir el archivo, convertirlo en una cadena para leerlo y, luego, convertir esta cadena en una lista almacenada en la variable `ip_addresses`. A continuación, iteré a través de las direcciones IP en `remove_list` y eliminé las direcciones IP de la lista `ip_addresses` con el método `.remove()`. Posteriormente, utilicé el método `.join()` para volver a convertir `ip_addresses` en una cadena para poder sobrescribir el contenido del archivo `"allow_list.txt"` con la lista revisada de direcciones IP.