Algorithm for file updates in Python

Project description

In our company, we manage access to sensitive data through a whitelist of IP addresses. This whitelist is maintained in a file named“allow_list.txt. Additionally, we have a separate list for removing IP addresses that should be denied access to this data. I've developed a method to automatically update the“allow_list.txt”file by removing the IP addresses that are no longer permitted access.

Open the file that contains the allow list

In the initial phase of the algorithm, I accessed the “allow_list.txt” file. To begin with, I set the filename as a string to the variable named import_file.

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

import_file = "allow_list.txt"
```

I used a “with” statement to open a file.

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

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

In my code, I utilize the with statement in conjunction with the .open() function in ‘read’ mode to access the allow list file. The goal of this operation is to retrieve the IP addresses contained within the file. The with keyword is beneficial as it automatically closes the file once the operations within the with block are completed. In the line with open(import_file, "r") as file:, the open() function takes two arguments. The first specifies the file to be opened, while the second,‘r’, signifies that the file is to be opened in read mode. The as keyword is used to assign the result of the .open() function to a variable named file, which is used within the with block.

Read the file contents

To interpret the contents of the file, I employed the .read() function, which transformed it into a string.

```
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()
```

By employing the .open() function with the “r” argument, which stands for “read”, I’m able to invoke the .read() function within the with statement. The .read() function transforms the file into a string, enabling me to read its contents. I applied the .read() function to the file variable, which was defined in the with statement. Subsequently, I assigned the output of this function, which is a string, to the ip_addresses variable. In essence, this code segment reads the “allow_list.txt” file and converts it into a string, which I can later manipulate and extract data from in my Python script.

Convert the string into a list

To facilitate the removal of specific IP addresses from the allow list, it was necessary for the data to be in a list format. Consequently, I employed the .split() function to transform the ip_addresses string into a list.

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

ip_addresses = ip_addresses.split()
```

The .split() method is invoked by attaching it to a string variable. Its function is to transform the string’s contents into a list. The rationale behind dividing ip_addresses into a list is to simplify the process of removing IP addresses from the allow list. The .split() method, by default, divides the text into list elements based on whitespace. In this algorithm, the .split() method processes the data in

the ip_addresses variable, which is a string of IP addresses each separated by a whitespace, and transforms this string into a list of IP addresses. This list is then stored by reassigning it back to the ip_addresses variable.

Iterate through the remove list

An essential component of my algorithm is the process of cycling through the IP addresses that constitute the elements in the remove_list. To accomplish this, I integrated a for loop into the code.

```
for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)
```

In Python, a for loop is used to execute a block of code repeatedly for a given sequence. The primary function of a for loop in a Python algorithm such as this one is to apply certain code instructions to every item in a sequence. The for keyword initiates the for loop, which is then followed by the loop variable element and the keyword in. The in keyword signifies that the loop should iterate over the ip_addresses sequence and assign each value to the loop variable element.

Remove IP addresses that are on the remove list

The objective of my algorithm is to eliminate any IP address from the ip_addresses allow list that is also present in the remove_list. Given that there were no duplicate entries in ip_addresses, I was able to accomplish this using the subsequent code:

```
for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)
```

Initially, inside my for loop, I established a condition to check if the loop variable element was present in the ip_addresses list. This was necessary because invoking .remove() on elements not existing in ip_addresses would lead to an error. Subsequently, within this condition, I applied the .remove() method to ip_addresses. I used the loop variable element as the argument, ensuring that each IP address present in the remove_list was eliminated from ip_addresses.

Update the file with the revised list of IP addresses

In the concluding phase of my algorithm, it was necessary to refresh the allow list file with the updated list of IP addresses. To achieve this, the first step was to transform the list back into a string format. I accomplished this using the .join() function.

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

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

The .join() function amalgamates all elements in an iterable into a single string. This function is invoked on a string that contains characters which will serve as separators between the elements in the iterable once they are combined into a string. In this particular algorithm, I utilized the .join() function to convert the ip_addresses list into a string, enabling me to use it as an argument for the .write() function when updating the “allow_list.txt” file. I employed the string (“ ”) as the separator, directing Python to position each element on a separate line.

Next, I employed an additional with statement along with the .write() function to refresh the file.

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

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

  file.write(ip_addresses)
```

On this occasion, I employed the “w” argument with the open() function within my with statement. This argument signifies my intention to open a file for overwriting its contents. When I use this “w” argument, I can invoke the .write() function within the with statement. The .write() function writes string data to a specified file, replacing any pre-existing content. In this instance, my goal was to write the updated allow list as a string to the “allow_list.txt” file.

As a result, any IP addresses that were removed from the allow list would no longer have access to the restricted content. To overwrite the file, I appended the .write() function to the file object that I identified in the with statement. I used the ip_addresses variable as the argument, indicating that the contents of the file specified in the with statement should be replaced with the data in this variable.

Summary

I devised an algorithm that eliminates IP addresses, specified in a remove_list variable, from the “allow_list.txt” file which contains authorized IP addresses. This algorithm entails opening the file, transforming it into a string for reading, and subsequently converting this string into a list that is stored in the ip_addresses variable. I then looped through the IP addresses in remove_list. During each iteration, I checked if the element was present in the ip_addresses list. If it was, I utilized the .remove() method to expunge the element from ip_addresses. Following this, I employed the .join() method to revert ip_addresses back into a string, enabling me to overwrite the “allow_list.txt” file with the updated list of IP addresses.