Project Description

In our organization, access to restricted content is managed through an "allow_list.txt" file that contains a list of approved IP addresses. Additionally, we have a separate list called the "remove list" that contains IP addresses that should no longer have access to the restricted content. To streamline the process of updating the "allow_list.txt" file and removing the unnecessary IP addresses, I have developed an algorithm that automates this task.

Open the file that contains the allow list

```
# Step 1: Open the file that contains the allow list
import_file = "allow_list.txt"
```

Next I used the with statement top open a file.

```
# Step 2: Read the file contents
with open(import_file, "r") as file:
    ip_addresses = file.read()
```

In my algorithm, I utilize the "with" statement along with the ".open()" function in read mode to access the allow list file. By opening the file using this method, I gain the ability to retrieve the IP addresses stored within it. The "with" keyword is essential as it ensures proper resource management by automatically closing the file once the code exits the "with" statement. The code snippet with open (import_file, "r") as file: demonstrates this usage, where the "open()" function takes two parameters. The first parameter specifies the file to be imported, while the second parameter, "r," indicates that the file is to be read. Moreover, the "as" keyword helps assign the variable named "file," which holds the output of the ".open()" function while I operate within the "with" statement.

```
# Step 3: Convert the string into a list
ip_addresses = ip_addresses.split()
```

To retrieve the file contents as a string, I employed the ".read()" method.

```
# Step 4: Iterate through the remove list
remove_list = ["192.168.1.100", "10.0.0.5", "172.16.0.1"]  # Replace with your actual remove_list
```

A crucial aspect of my algorithm entails iterating through the IP addresses listed in the remove_list. To accomplish this, I utilized a for loop:

```
for element in remove_list:
```

In Python, a for loop is designed to execute a set of code statements repeatedly for each element in a given sequence. The primary purpose of using a for loop in a Python algorithm is to apply specific actions to every item within the sequence. The for loop begins with the "for" keyword, followed by the loop variable (in this case, "element"), and the "in" keyword, which signifies iterating through the sequence "ip_addresses" and assigning each value to the loop variable "element."

Remove IP Address that are on Remove List

To execute my algorithm successfully, I need to eliminate any IP addresses present in the "ip_addresses" allow list that also appear in the "remove_list". Since there are no duplicate entries in "ip_addresses", I employed the following code to accomplish this task:

```
# Step 5: Remove IP addresses that are on the remove list
if element in ip_addresses:
    ip_addresses.remove(element)
```

In the initial part of my for loop, I established a condition to check whether the loop variable "element" exists in the "ip_addresses" list. This step is crucial to avoid errors when using the .remove() method on elements that are not present in "ip_addresses".

Subsequently, within the same conditional block, I executed the .remove() method on "ip_addresses". By passing the loop variable "element" as the argument, I ensured that every IP address present in the "remove_list" was removed from the "ip_addresses" list.

Update the file with the revised list of IP addresses

As the last step of my algorithm, I had to update the "allow_list.txt" file with the modified list of IP addresses. To achieve this, I converted the list back into a string format, employing the .join() method.

```
# Step 6: Update the file with the revised list of IP addresses
updated_contents = "\n".join(ip_addresses)
```

The .join() method in Python combines all elements of an iterable into a single string. It takes a string as an argument, which will be used as the separator to join the elements together in the resulting string. In this algorithm, I utilized the .join() method to convert the list "ip_addresses" into a string format, which was then passed as an argument to the .write() method when updating the "allow_list.txt" file. To ensure each element is written on a new line, I used the "\n" string as the separator.

Following this, I employed another with statement along with the .write() method to update the file with the revised list of IP addresses. The "with" statement ensures that the file is properly handled and closed after the operation is completed.

```
# Write the updated contents back to the file
with open(import_file, "w") as file:
    file.write(updated_contents)
```

This time, I utilized a second argument of "w" with the open() function within the "with" statement. This "w" argument indicates that I intend to open the file to overwrite its existing contents. By using "w", I was able to call the .write() function within the "with" statement, which writes the string data to the specified file, effectively replacing any previous content.

In this specific scenario, my objective was to write the updated allow list as a string to the file "allow_list.txt". This action ensures that any IP addresses removed from the allow list will no longer have access to the restricted content. To achieve this, I appended the .write() function to the file object "file" that I previously identified in the "with" statement. I passed the "ip_addresses" variable as the argument to specify that the data in this variable should replace the contents of the file specified in the "with" statement.

Summary

I developed an algorithm to update the "allow_list.txt" file by removing IP addresses listed in the "remove_list" variable. The process involved opening the file, converting its content into a string for reading, and then converting that string into a list stored as "ip_addresses." I proceeded to iterate through the IP addresses in "remove_list," checking if each element existed in the "ip_addresses" list. If it was present, I used the .remove() method to eliminate it from "ip_addresses." Finally, I employed the .join() method to convert the modified

"ip_addresses" list back into a string and overwrite the contents of the "allow_list.txt" file with the updated list of IP addresses.