# Algorithm for file updates in Python

## Project description

At my organization, access to restricted content is managed through an allow list of IP addresses stored in a file called **"allow\_list.txt"**. This file contains the IP addresses of employees who are permitted to access sensitive data. Periodically, I receive a **remove list**—a list of IP addresses that should no longer have access.

To streamline this process, I developed a Python algorithm to automatically update the **"allow\_list.txt"** file by removing any IP addresses found in the remove list. This ensures that only authorized personnel can access restricted content

## Open the file that contains the allow list

To begin, I needed to access the **"allow\_list.txt"** file so that I could modify its contents. I first assigned the filename to a variable:



Next, I used a **with** statement to open the file:



The **with** statement ensures that the file is properly closed after it is read. The "r" argument in the **open()** function specifies that I am opening the file in read mode. I assigned the contents of the file to a variable called **ip\_addresses**, which allows me to process the data in the next steps.

## Read the file contents

After opening the file, I needed to extract its contents in a way that allowed me to manipulate the data. Since the file contains multiple IP addresses stored as a space-separated string, I used the **.read()** method to convert the file’s contents into a single string:



At this point, **ip\_addresses** was a string containing all IP addresses in the allow list. My next step was to convert this string into a list so that I could remove specific IP addresses.

## Convert the string into a list

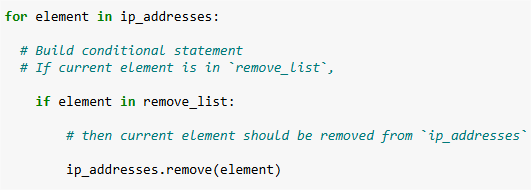
Since I needed to remove individual IP addresses from the allow list, I used the **.split()** method to convert the string into a list:



The **.split()** method divides a string into a list based on whitespace. This transformation was necessary because it allowed me to process each IP address as a separate element, making it easier to remove unwanted entries.

## Iterate through the remove list

The next step was to go through the **remove list** and check if any of its IP addresses were present in the allow list. To accomplish this, I used a **for** loop:



### **How This Works:**

* The **for** loop goes through each IP address in the **remove list**.
* If an IP address is found in the **allow list**, it is removed using the **.remove()** method.

This ensures that any IP address appearing in both lists is deleted from **ip\_addresses**.

## Remove IP addresses that are on the remove list

For each IP in the remove list, I check if it exists in the allow list. If it does, I remove it to ensure that only authorized employees retain access.



First, within my for loop, I created a conditional that evaluated whether or not the loop variable element was found in the ip\_addresses list. I did this because applying .remove() to elements that were not found in ip\_addresses would result in an error.

Then, within that conditional, I applied .remove() to ip\_addresses. I passed in the loop variable element as the argument so that each IP address that was in the remove\_list would be removed from ip\_addresses.

## Update the file with the revised list of IP addresses

After removing the unwanted IP addresses, I needed to save the updated list back to **"allow\_list.txt"**. Since the file originally contained IP addresses as a space-separated string, I used the **.join()** method to convert the list back into a string before writing it to the file:



Next, I used another **with** statement to overwrite the file with the updated data:



This time, I opened the file in **write mode ("w")**, which replaces its contents with the new data. The **.write()** method ensures that the revised allow list is saved correctly.

## Summary

I created an algorithm to automate the removal of unauthorized IP addresses from the "allow\_list.txt" file. First, I opened the allow list file using a with statement and read its contents, storing them as a string. Then, I converted the string into a list to allow easier manipulation. I iterated through the remove list to identify and delete any matching IP addresses. After that, I updated the file with the revised allow list by converting the list back into a string using the .join() method and writing the updated data back into the file. This process ensures that the allow list remains accurate and up to date, preventing unauthorized access to restricted content. By automating the update process, I have made it more efficient and less prone to human error.