# Algorithm for file updates in Python

## Project description

Access to restricted content is managed through an allow list of IP addresses stored in the "allow\_list.txt" file. A separate remove list identifies IP addresses that should be revoked. I developed an algorithm to automate updating the allow list by removing IP addresses that no longer require access.

## Open the file that contains the allow list

For the initial step of the algorithm, I opened the "allow\_list.txt" file. First, I assigned the file name as a string to the import\_file variable.



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



In my algorithm, the with statement is combined with the .open() function in read mode ("r") to access the allow list file. The purpose of opening this file is to retrieve the IP addresses stored within it. The with keyword helps manage resources efficiently by automatically closing the file once the block is exited.

The statement with open(import\_file, "r") as file: contains two parameters within the open() function. The first specifies the file to be opened, while the second ("r") indicates that the file is being opened in read mode. Additionally, the as keyword assigns the file object to the variable file, which stores the output of the .open() function while working within the with block.

## Read the file contents

To read the file's contents, I used the .read() method to convert it into a string.



When opening a file with the "r" argument for reading, I can call the .read() method within the with statement. This method reads the entire file and converts its contents into a string. I applied the .read() method to the file variable defined in the with statement and then assigned its output to the ip\_addresses variable.

In summary, this code reads the contents of the "allow\_list.txt" file into a string, enabling me to later process, organize, and extract relevant data within my Python program.

## Convert the string into a list

To remove individual IP addresses from the allow list, I first needed to convert the data into a list format. To achieve this, I used the .split() method to transform the ip\_addresses string into a list.



The .split() method is applied directly to a string variable and functions by breaking the string into a list of elements. The purpose of splitting ip\_addresses into a list is to facilitate the removal of specific IP addresses from the allow list. By default, .split() separates the string based on whitespace, creating individual list elements.

In this algorithm, the .split() method processes the ip\_addresses string, which contains IP addresses separated by whitespace, and converts it into a list of individual addresses. The resulting list is then reassigned to the ip\_addresses variable for further manipulation.

## Iterate through the remove list

A crucial aspect of my algorithm is iterating through the IP addresses in the remove\_list. To achieve this, I used a for loop.



In Python, a for loop is used to execute code repeatedly for each element in a given sequence. The purpose of using a for loop in this algorithm is to apply specific operations to all elements in the sequence. The loop begins with the for keyword, followed by a loop variable (element) and the in keyword. The in keyword signals the loop to iterate through the sequence (remove\_list), assigning each IP address to the element variable during each iteration

## Remove IP addresses that are on the remove list

My algorithm needed to remove any IP address from the ip\_addresses list that was also present in remove\_list. Since ip\_addresses did not contain duplicates, I was able to achieve this with the following approach:



First, inside my for loop, I added a conditional statement to check whether the loop variable element was in the ip\_addresses list. This check was necessary because using .remove() on an item that isn’t in the list would cause an error.

Next, within the conditional, I applied the .remove() method to ip\_addresses, passing element as the argument. This ensured that each IP address found in remove\_list was successfully removed from ip\_addresses.

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

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



The .join() method merges all elements of an iterable into a single string. It is applied to a string that specifies the separator between the elements. In this case, I used "\n" as the separator to ensure each IP address appeared on a new line in the output string. This allowed me to pass the formatted string as an argument to the .write() method when updating "allow\_list.txt".

Next, I used another with statement along with the .write() method to update the file:



This time, I opened the file using the "w" argument in the open() function, which indicates that the file should be opened in write mode, replacing its existing content. Within the with block, I called the .write() method on the file object, passing in the ip\_addresses string. This ensured that the updated allow list was written to "allow\_list.txt", effectively removing any IP addresses that were no longer permitted.

## Summary

I developed an algorithm to remove IP addresses from the "allow\_list.txt" file based on those listed in remove\_list. The process involved reading the file’s contents, converting them into a list (ip\_addresses), and iterating through remove\_list to remove any matching entries. After updating the list, I used the .join() method to convert it back into a string and overwrite the file with the revised allow list.