# Algorithm for file updates in Python

## Project description

In our organization, we manage access to restricted content using an allow list of IP addresses, which is stored in the “allow\_list.txt” file. There is also a separate remove list that contains IP addresses that should be revoked from accessing this content. To streamline this process, I have developed an algorithm that automatically updates the “allow\_list.txt” file by removing the IP addresses listed in the remove list.

## Open the file that contains the allow list

Firstly, I accessed the “allow\_list.txt” file. To do this, I assigned the file name, represented as a string, to the variable named import\_file.



Next, I used a ‘with’ statement to open the file.



In my algorithm, I employ the ‘with’ statement in conjunction with the .open() function in read mode to access the allow list file. The objective of this action is to retrieve the IP addresses contained within the allow list file. The ‘with’ keyword aids in resource management by ensuring the file is closed once we exit the ‘with’ statement. In the code segment ‘with open(import\_file, “r”) as file:’, the open() function has two parameters. The first specifies the file to be imported, while the second defines the intended operation on the file. Here, “r” signifies that the file is to be read. The ‘as’ keyword is also used in the code to assign a variable named ‘file’. This variable temporarily holds the output of the .open() function while operations are performed within the ‘with’ statement.

## Read the file contents

I used the .read() method to convert it into string so I could read the file content.



When the .open() function is used with the “r” argument for “read”, it allows me to call the .read() function within the body of the ‘with’ statement. The .read() method transforms the file into a string, enabling me to read its contents. I applied the .read() method to the ‘file’ variable specified in the ‘with’ statement, and then assigned the string output of this method to the ‘ip\_addresses’ variable.

In essence, this code reads the contents of the “allow\_list.txt” file and converts it into a string format. This allows me to subsequently manipulate and extract data from this string in my Python program.

## Convert the string into a list

To remove specific IP addresses from the allow list, it was necessary for me to have the data in a list format. Consequently, I utilized the .split() method to transform the ‘ip\_addresses’ string into a list.



The .split() function is invoked by appending it to a string variable. Its role is to transform the contents of a string into a list. The reason for splitting ‘ip\_addresses’ into a list is to facilitate the removal of IP addresses from the allow list. By default, the .split() function divides the text by whitespace into list elements. In this algorithm, the .split() function processes the data stored in the ‘ip\_addresses’ variable, which is a string of IP addresses each separated by a whitespace, and converts this string into a list of IP addresses. This list is then reassigned back to the ‘ip\_addresses’ variable.

## Iterate through the remove list

A crucial aspect of my algorithm involves cycling through the IP addresses that are elements in the remove\_list. To achieve this, I integrated a ‘for’ loop into the code.



The ‘for’ loop in Python executes a block of code for a specified sequence. The primary function of the ‘for’ loop in a Python algorithm like this one is to apply certain code statements to all elements within a sequence. The loop begins with the ‘for’ keyword, followed by the loop variable ‘element’ and the ‘in’ keyword. The ‘in’ keyword signifies that the iteration should occur through the ‘ip\_addresses’ sequence, assigning each value to the loop variable ‘element’.

## Remove IP addresses that are on the remove list

The algorithm I’ve designed necessitates the removal of any IP address from the ‘ip\_addresses’ allow list that also includes the ‘remove\_list’. Given that there were no duplicates in ‘ip\_addresses’, I was able to accomplish this using the following code:



Initially, within my ‘for’ loop, I established a condition to evaluate whether the loop variable ‘element’ was present in the ‘ip\_addresses’ list. This step was crucial because attempting to apply .remove() to elements not found in ‘ip\_addresses’ would result in an error.

Subsequently, within that conditional, I applied the .remove() method to ‘ip\_addresses’. I passed in the loop variable ‘element’ as the argument, ensuring that each IP address present in the ‘remove\_list’ would be eliminated from ‘ip\_addresses’.

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

In the final stage of my algorithm, I had to refresh the allow list file with the updated set of IP addresses. This required me to transform the list back into a string format. I accomplished this using the .join() method.



The .join() method combines all items in an iterable into a single string. This method is applied to a string that contains characters which will act as separators between the elements in the iterable once they are combined into a string. In this procedure, I utilized the .join() method to form a string from the list ip\_addresses so that it could be used as an argument for the .write() method when writing to the file “allow\_list.txt”. I employed the string (“\n”) as the separator to instruct Python to position each element on a separate line.

Subsequently, I used another with statement and the .write() method to refresh the file:



This time, I employed a second argument of “w” with the open() function in my with statement. This argument signifies that I intend to open a file to overwrite its contents. When using this “w” argument, I can invoke the .write() function within the body of the with statement. The .write() function writes string data to a specified file and overrides any existing file content. In this instance, I aimed to write the updated allow list as a string to the file “allow\_list.txt”. Consequently, the restricted content will no longer be accessible to any IP addresses that were eliminated from the allow list. To rewrite the file, I added the .write() function to the file object file that I identified in the with statement. I input the ip\_addresses variable as the argument to specify that the contents of the file defined in the with statement should be supplanted with the data in this variable.

## Summary

I wrote an algorithm that eliminates IP addresses specified in a remove\_list variable from the “allow\_list.txt” file of authorized IP addresses. This algorithm entailed opening the file, transforming it into a string for reading, and then converting this string into a list stored in the ip\_addresses variable. I then iterated through the IP addresses in remove\_list. During each iteration, I assessed whether the element was part of the ip\_addresses list. If it was, I employed the .remove() method on it to delete the element from ip\_addresses. Subsequently, I used the .join() method to revert the ip\_addresses back into a string so that I could overwrite the contents of the “allow\_list.txt” file with the updated list of IP addresses.