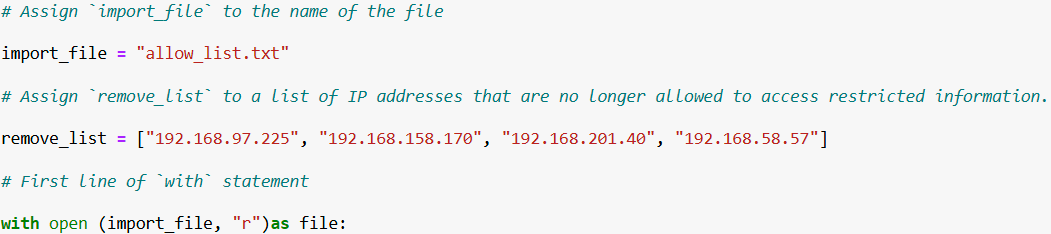
# Algorithm for file updates in Python

## Project description

In this scenario, as a security professional at my organization, access to restricted content is controlled in which only certain ip addresses can access it, these ip addresses are stored in the “allow\_list.txt” file. A separate list of ip addresses are specified in a remove\_list where it contains the ip\_addresses which does have access to the restricted content, I developed an algorithm to update the “allow\_list.txt”file and remove the ip addresses which does not have access automatically.

## Open the file that contains the allow list

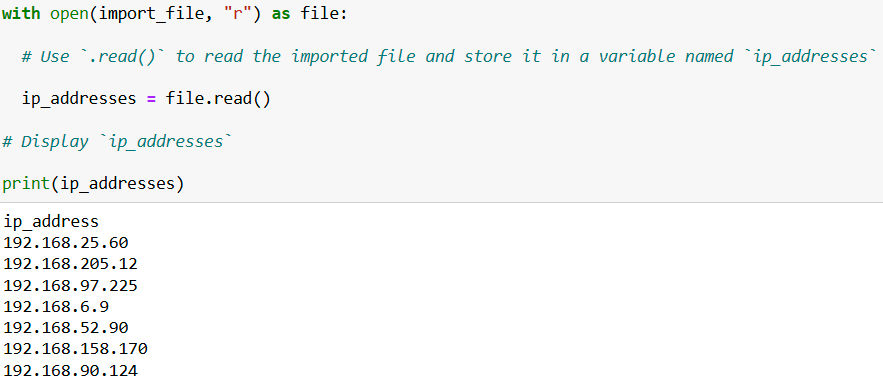


To open the allow\_list.txt file, I first assigned the file as a string to a variable named import\_file.

Then I opened the file using the with statement.

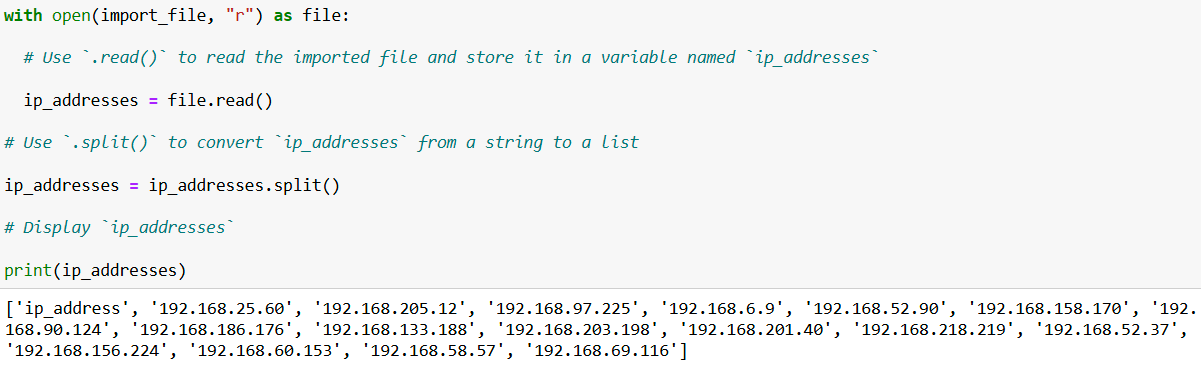
Here I use the with statement along with the .open() function in read mode to access the allow list file and retrieve the IP addresses stored in it. The with keyword ensures proper resource management by automatically closing the file once the code block is executed. In the line with open(import\_file, "r") as file:, the open() function takes two parameters: the first specifies the file name, and the second specifies the mode, which is "r" for read in this case. The as keyword assigns the file to the variable file, which holds the file's content while I'm working within the with statement.

## Read the file contents



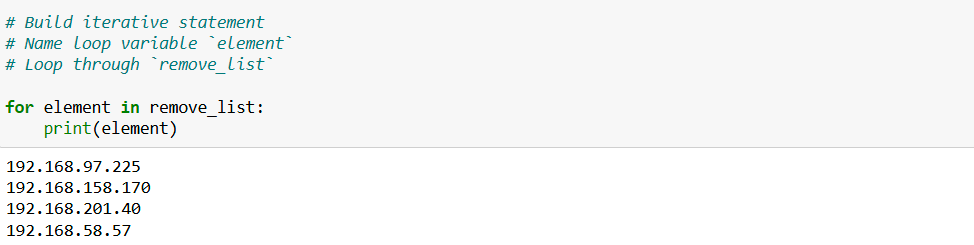
When using the .open() function with the argument "r" for "read," I can call the .read() function within the with statement. The .read() method converts the file content into a string, allowing me to read it. I applied the .read() method to the file variable specified in the with statement and assigned its string output to the variable ip\_addresses.

## Convert the string into a list



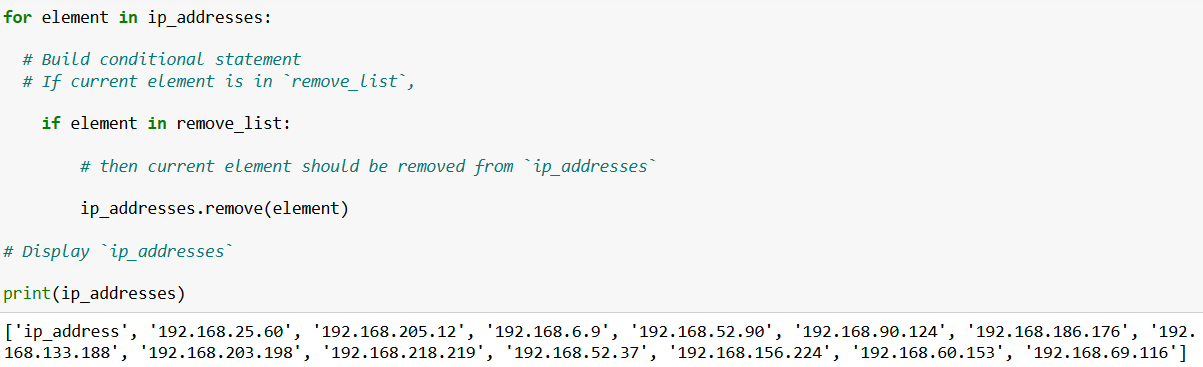
The .split() function is used by appending it to a string variable. It converts the contents of the string into a list. Splitting ip\_addresses into a list makes it easier to remove IP addresses from the allow list. By default, the .split() function splits the text by whitespace into list elements. In my algorithm, the .split() function takes the string of IP addresses stored in the variable ip\_addresses, which are separated by whitespace, and converts it into a list of IP addresses. I then reassign this list back to the variable ip\_addresses.

## Iterate through the remove list



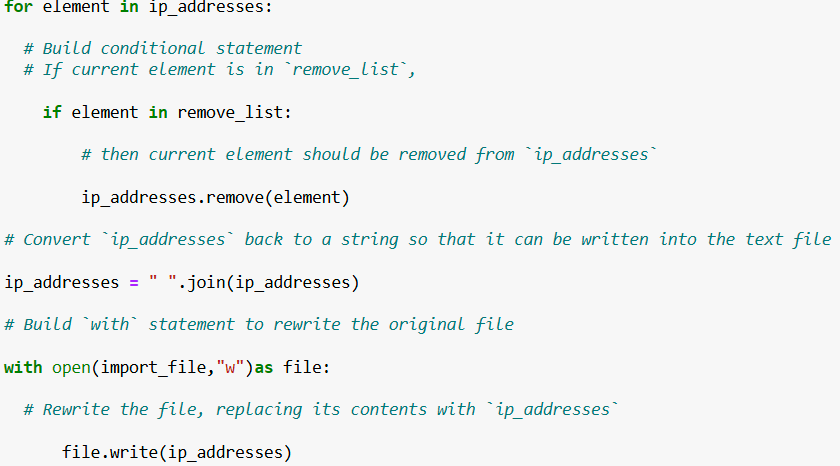
The for loop in Python is used to repeat code for a specified sequence. For this algorithm, its main purpose is to apply certain code statements to every element in the sequence. The loop begins with the for keyword, followed by the loop variable element and the keyword in. The in keyword signals that the loop should iterate through the sequence of ip\_addressesin remove list, assigning each value to the loop variable element.

## Remove IP addresses that are on the remove list



First, within my for loop, I created a conditional to check if the loop variable element was found in the ip\_addresses list. This was necessary to avoid errors from applying .remove() to elements not present in ip\_addresses. Then, within that conditional, I used .remove() on ip\_addresses, passing in the loop variable element as the argument. This ensured that each IP address in the remove\_list was removed from ip\_addresses.

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



Finally, 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 combines all items in an iterable into a single string. It's applied to a string that contains the characters which will separate the elements in the iterable once they are joined into a string. In my algorithm, I used the .join() method to create a string from the ip\_addresses list. This string could then be passed as an argument to the .write() method when writing to the file "allow\_list.txt". I used the string ("\n") as the separator to ensure each element would be placed on a new line.

Next, I used another with statement and the .write() method to update the file. This time, I used "w" as the second argument with the open() function in my with statement. This indicates that I want to open the file to write over its contents. Using the "w" argument, I could call the .write() function within the body of the with statement. The .write() function writes string data to a specified file and replaces any existing content. In this case, I wanted to write the updated allow list as a string to the file "allow\_list.txt". This way, the restricted content will no longer be accessible to any IP addresses that were removed from the allow list. To rewrite the file, I appended the .write() function to the file object file specified in the with statement, passing in the ip\_addresses variable as the argument to ensure the file's contents were replaced with the data from this variable.

## Summary

I developed an algorithm to remove IP addresses listed in a `remove\_list` from the "allow\_list.txt" file, which contains approved IP addresses. Here's how it works:

1. Open and Read the File: I started by opening the file and converting its contents into a string. This string was then transformed into a list, stored in the variable `ip\_addresses`.

2. Remove IP Addresses: I iterated through each IP address in the `remove\_list`. For each address, I checked if it was in the `ip\_addresses` list. If it was, I used the `.remove()` method to delete it from the list.

3. Update the File: Finally,After removing the addresses, I used the `.join()` method to convert the updated `ip\_addresses` list back into a string. This string was then written over the existing content of the "allow\_list.txt" file, ensuring that the file now only includes the revised list of approved IP addresses.