# Algorithm for file updates in Python

## Project description

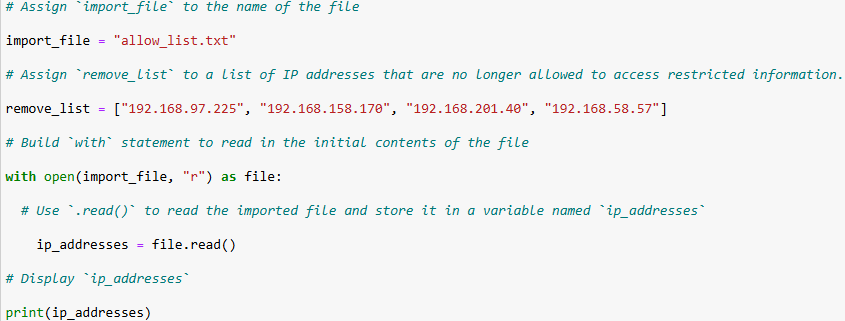
In this project, I wrote a Python script to manage an allow list of IP addresses for a healthcare company’s secure network. The goal was to automatically remove any IPs that were on a separate remove list, so only authorized employees could access sensitive data. I used Python to open the allow list file, read its contents, and turn it into a list that I could work with. Then I checked each IP against the remove list and updated the original file with the revised list. This project shows how Python can make routine cybersecurity tasks faster and more accurate.

## Open the file that contains the allow list



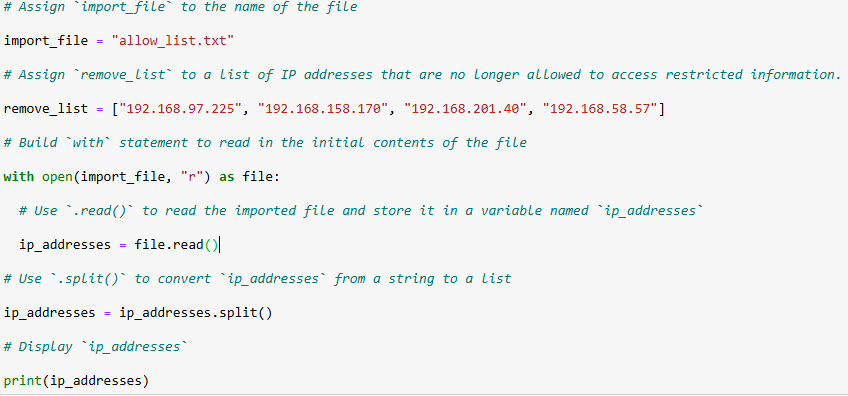
Using the with open statement, we open the file specified by the import\_file variable in read mode and assign it to the variable file for use within the block.

## Read the file contents



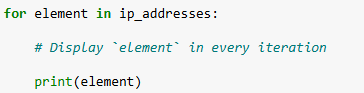
Within the with open block, we use the .read() method on the new file variable to read the import\_file argument and assign it to the variable ip\_addresses. We then use print() to show the new variable ip\_addresses - the content of the file will show as a string.

## Convert the string into a list



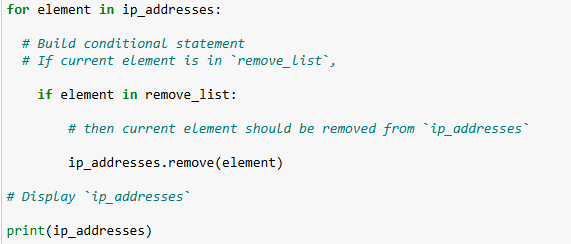
Next, we apply the .split() method to the ip\_addresses string. We reassign the result back to the variable ip\_addresses, so it now stores a list instead of a string. We then call print() to display ip\_addresses, which now shows each IP address as a separate element inside a list.

## Iterate through the remove list



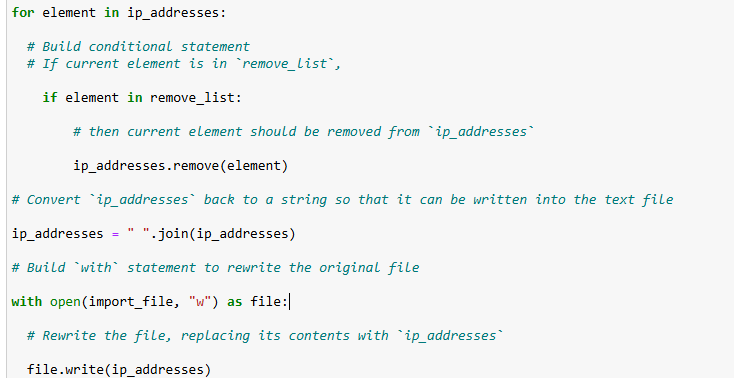
We then create a for loop using the element variable to parse through the ip\_addresses list from the last step. We use print() to display each element in the ip\_addresses list

## Remove IP addresses that are on the remove list



Inside the for loop, we add an if statement that checks whether the current loop variable (element) is also present in the remove\_list. If the condition evaluates to True, we call the .remove() method on ip\_addresses and remove that matching element. This ensures that any IP address appearing in both lists is deleted from the allow list.

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

  
After removing the IP addresses in the remove\_list, we must convert the updated ip\_addresses list back into a single string before writing it to the file. We do this by calling the .join() method on the string " ", which inserts a space between each element of the list as they are combined into one continuous string.

Next, we use a with open statement and specify the mode "w" (write mode) to open the file referenced by import\_file. Inside this block, we call the .write() method to overwrite the file’s contents with the new ip\_addresses string. This step updates the original file so that it reflects the revised allow list.

## Summary

The algorithm starts by opening the allow list file using a with open statement and reading its contents with .read(). I turned the contents into a list with .split() so each IP address could be handled individually. Then, I looped through the list and checked each IP against the remove list. If an IP was on the remove list, I removed it from the main list with .remove(). After cleaning up the list, I joined it back into a string with .join() and wrote it back to the original file using .write(). This process ensures the allow list always stays up-to-date while automating what would otherwise be a manual, error-prone task.