# `Algorithm for file updates in Python

## Project description

As a security professional that is working at a healthcare company. I am tasked with regularly updating a allow list with employee IP’s that are granted access to restricted content. And a remove list of restricted employee IP’s that are to be removed from the allow list.

My task is to create an algorithm that uses Python code to check whether the allow list contains any IP address identified on the remove list. If so, I should remove the IP addresses from the file containing the allow list.

## Open the file that contains the allow list

To open the allow\_list.txt file that contains the allowed employee IP’s you need to follow the code bellow



1. Firstly we assign a variable with the name import\_file to the allow\_list.txt.

import\_file = "allow\_list.txt"

1. Then we add the remove\_list that is also a variable containing the IP’s that are to be removed.

remove\_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

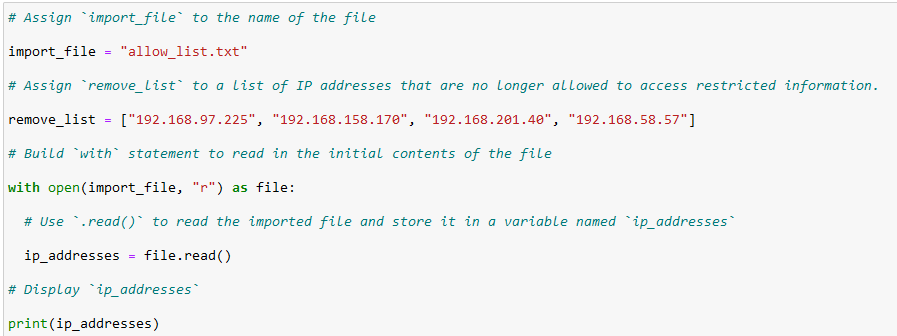
1. Lastly we open the file with the with statement using the open() function with the “r” parameter and save it as a file variable.

with open(import\_file, "r") as file:

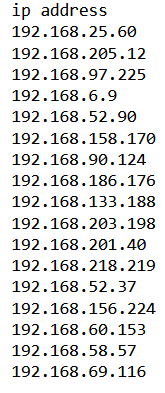
## Read the file contents

To read the file contents we make a new variable with the name ip\_addresses and pass the value of file.read(). Our allow list is contained in the file variable and we add the .read() method to read the file.

ip\_addresses = file.read()



We use the print (ip\_addresses) to display the IP’s

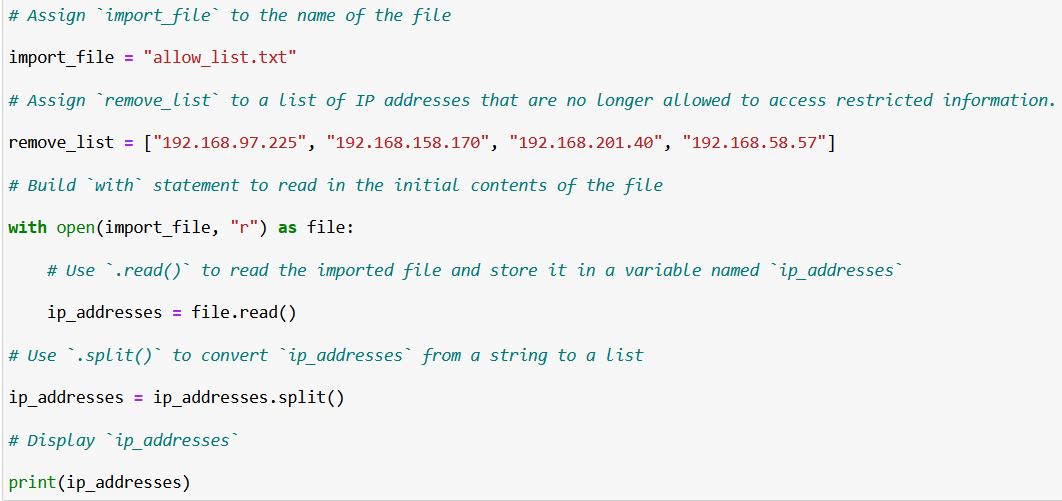


## Convert the string into a list

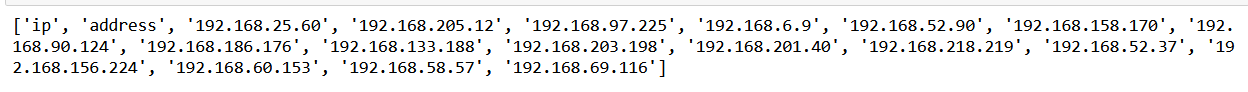
In order to remove the individual IP addresses from the allow list, The IP addresses need to be in a list format.

So we need to update the ip\_addresses variable and use the .split() method to convert the string variable into a list.

ip\_addresses = ip\_addresses.split()



Ip\_addresses updated as a list:

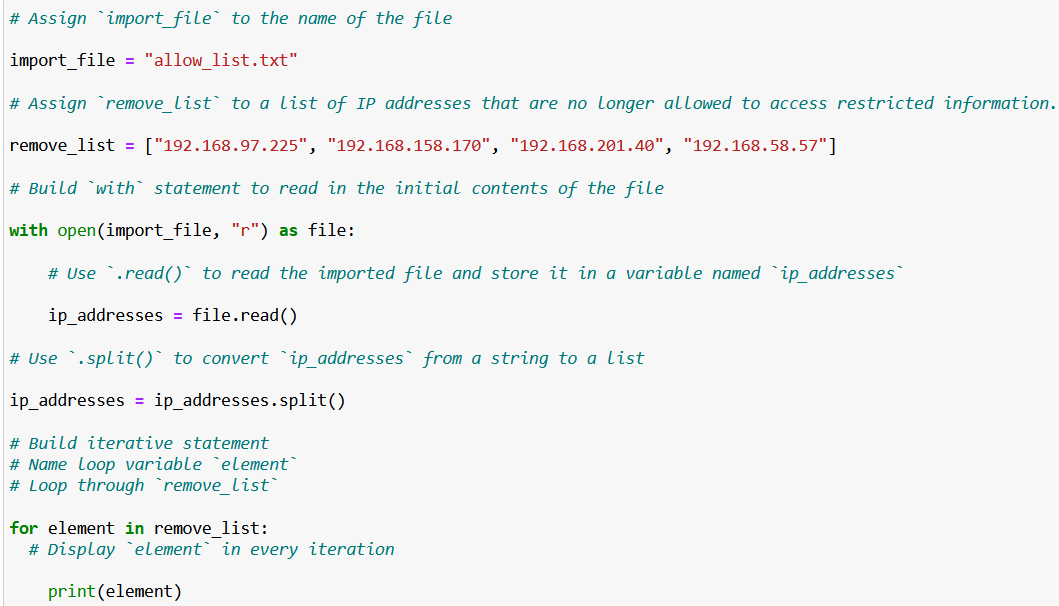


## Iterate through the remove list

The remove\_list is called containing the IP addresses that should be removed from the ip\_addresses list.

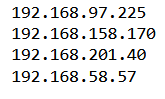
To do so we use a for loop that will loop through the remove\_list and we use element as the loop variable.

for element in remove\_list:



The for loop gets a single IP from the remove\_list and stores it in the variable element ,the loop is repeated until every IP is passed in the element variable once.

If we print the element in the for loop every passed ip will be printed once



## Remove IP addresses that are on the remove list

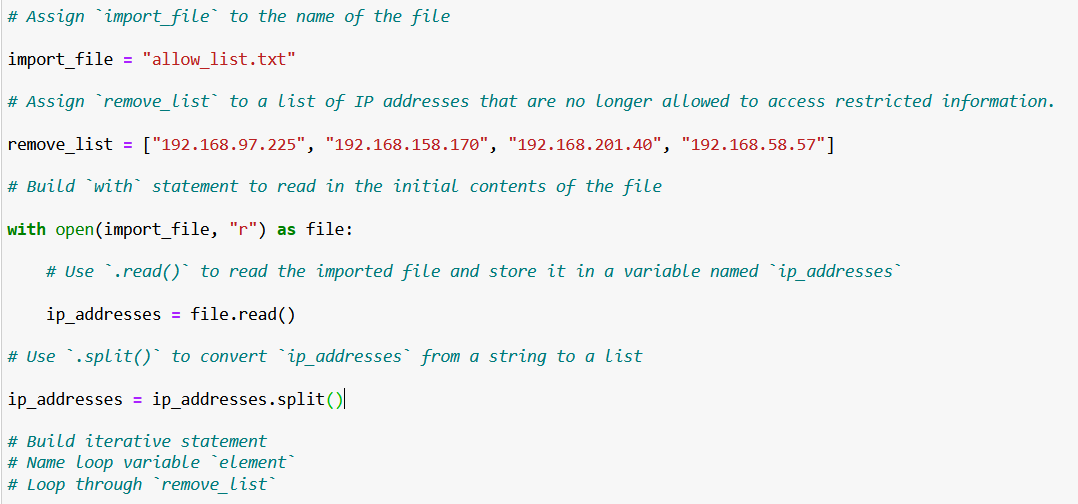
In the body of the iterative statement, we add code that will remove all the IP addresses from the allow list that are also on the remove list.

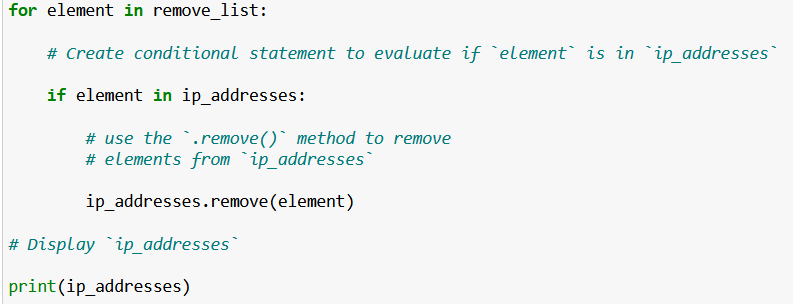
First we create a conditional that evaluates if the loop variable element is part of the Ip\_addresses list.

If element in ip\_addresses

Then,within that conditional,we apply the .remove() method to the ip\_addresses list and remove the IP addresses identified in the loop variable element.

Ip\_addresses.remove(element)





We print the ip\_addresses to see if we have removed the remove\_list IP’s



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

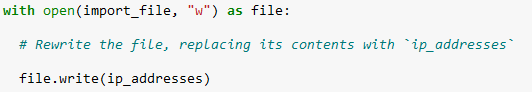
Now that we have removed the IP addresses from the ip\_address variable, we can complete the algorithm by updating the file with the revisited list. To do so, we must first convert the ip\_addresses list back into a string using the .join() method. Apply .join() to the string “\n” in order to separate the elements in the file by placing them on a new line.

Ip\_addresses = “\n”.join(ip\_addresses)

Then, we use another with statement and the .write() method to write over the file assigned to the import\_file variable.

with open(import\_file, “w”) as file:



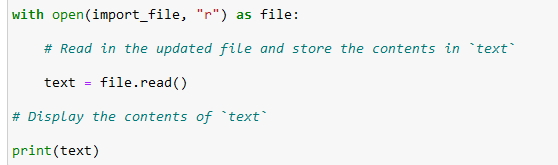


To read the updated file we need to add

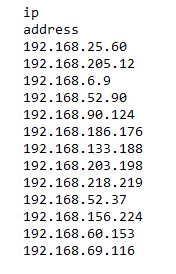
with open(import\_file, “r”) as file:

To read the updated file and store the contents in ‘text’

text = file.read()



The displayed contents



We can see that the file has been rewritten.

## Summary

1. **Data Preparation**: Load the content of the allow list and remove list from their respective files into Python data structures (e.g., lists or sets) for easy manipulation.
2. **Iterative Comparison**: Iterate through each IP address in the remove list. For each IP address, check if it exists in the allow list data structure using a membership check operation
3. **Update Allow List**: If an IP address from the remove list is found in the allow list, remove it from the allow list data structure. This can be done by filtering the allow list data structure to exclude the identified IP address.
4. **Write Back to File**: Once all necessary updates have been made to the allow list data structure, write the updated IP addresses back to the allow list file, overwriting the previous content.