# Algorithm for file updates in Python

## Project description

As a security analyst working in a health care company, it is part of my responsibilities to keep the list of employees allowed to access SPII and PII updated. This list is called allow\_list.txt. There is also a list of employees that no longer need to have that access to restricted content, which is called remove\_list.txt. The employees are identified by their Ip addresses in both cases.

As this a regularly reoccurring event I have developed an algorithm that automatizes this task. See below.

## Open the file that contains the allow list

First step is assigning a string with the name “allow\_list.txt” to the import file variable. Next, I open it using with and store the file in a variable called file. I also create the “remove\_list” with a list of 4 ip addresses given to me that should no longer have restricted access. The with keyword helps manages resources by closing the file after exiting with the with the with statement.

See attached screenshot: “Open the file that contains the allow list.png”.

## Read the file contents

To be able to see the allow\_list I needed to convert the file to a string, which I did using the read() method and saved it as the variable ip\_addresses.

See attached screenshot: “Read the file contents.png”

## Convert the string into a list

As could be seen - print(ip\_addresses) - the ip addresses were not in a list format, which they need to be if individual ip addresses are to be removed. The command ip\_addresses.split() corrects this and set it up for the next step.

See attached screenshot: “Convert the string into a list.png”

## Iterate through the remove list

To access each ip address in the variable ip\_address and compare it the remove\_list a for loop with a nestled if statement was needed. That looked like this:

for element in ip\_addresses:

if element in remove\_list:

#do something

## Remove IP addresses that are on the remove list

If above code finds a match in the remove\_list the syntax ip\_addresses.remove(element) will remove that particular ip address from the list of ip\_addresses. Using the remove() method in this way works because we know there are no duplicates in the allow\_list.

See attached screenshot for the entire loop: “Remove IP addresses that are on the remove list.png”

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

With the previous step, the list was updated. But for the actual file to also be updated the list needed to be converted back into a string. I wanted each ip address to be on a separate line. Another with using write() then overwrites the file assigned in the import\_file variable. The final code snippet is this:

ip\_addresses = "\n".join(ip\_addresses)

# Build `with` statement to rewrite the original file

with open("allow\_list", "w") as file:

# Rewrite the file, replacing its contents with `ip\_addresses`

file.write("ip\_addresses")

A final clarification is that .join combines all items in an iterable into a string, and that string could then be passed as an argument into the write() method.

## Summary

I created an algoritm that removed any IP address specified in “remove\_list.txt” from the official “allow\_list.txt” of approved IP addresses. It consisted of several steps, the first of which involed opening the file, converting it to a string for readability and then to a list stored in a varibale called ip\_addresses. By iterating over this list and compare it value for value in the remove\_list, I evaluated if that ip address was also present in the remove\_list. If so, it was removed, Finally I reversed the process in the beginning of this summery: conversion back into a string so that I could overwrite the contents of the allow\_list.txt with the updated list of approved IP addresses.