# Algorithm for file updates in Python

Benjamin Taylor – Google Cybersecurity

## Project description

Access to restricted content is controlled with an allow list containing IP addresses. The "allow_list.txt" file holds these IP addresses. A separate remove list contains the IP addresses that shouldn't have access to the content anymore. I created an algorithm to automate updating the "allow_list.txt" file and remove the IP addresses that shouldn't have access.

## Open the file that contains the allow list

First, I opened the "allow_list.txt" file. I assigned this file name as a string to the variable "import_file":

```
import_file = “allow_list.txt”
```

Next, I used a "with" statement to open the file:

```
with open(import_file, “r”) as file:
```

The purpose of opening the file is to let me find the IP addresses that are stored in the allow list file.

## Read the file contents

To read the contents of the file, I used the ".read()" method to convert the contents to a string:

```
with open(import_file, “r”) as file:
      Ip_addresses = file.read()
```

The read method converts the file into a string and allows me to read it. I then assigned the string output of this method to the variable "ip_addresses".

## Convert the string into a list

To remove the individual IP addresses from the allow list, it had to be in listformat. I used the ".split()" method to convert the ip_addresses string into a list:

```
ip_addresses = ip_addresses.split()
```

The split function is called by appending it to a string variable, and converts the contents of a string into a list.

## Iterate through the remove list

To go through the IP address elements in the “remove_list”, I incorporated a “for” loop:

```
for element in remove_list:
```

The purpose of this “for” loop is to apply specific code statements to all elements in the sequence, in this case “remove_list”.

## Remove IP addresses that are on the remove list

To remove any duplicates in the “remove_list”, I just furthered my “for” loop:

```
for element in remove_list:
    if element in ip_addresses:
        ip_addresses.remove(element)
```

I created a conditional that checked whether or not “element” was found in the ip_addresses list. I did this so that there were no errors thrown, as applying “.remove()” where there are not any elements found would result in an error.

In this conditional, I applied “.remove()” to “ip_addresses”, so that each IP address found in the “remove_list” would be removed from “ip_addresses”.

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

Finally, I had to update the file with the revised list of addresses. I used the “.join()” method:

```
ip_addresses = “\n”.join(ip_addresses)
with open(import_file, “w”) as file:
    file.write(ip_addresses)
```

This code snippet shows how I joined the ip addresses together using line separation (“\n”), and then opened the file with writing permission to write the updated list.

## Summary

This algorithm ultimately removed IP addresses that were noted in the “remove_list” variable from the “allow_list.txt” file of approved IP addresses. This algorithm involved opening the le, converting it to a string to be read, and then converting this string to a list stored in the variable ip_addresses. I then iterated through the IP addresses in remove_list.