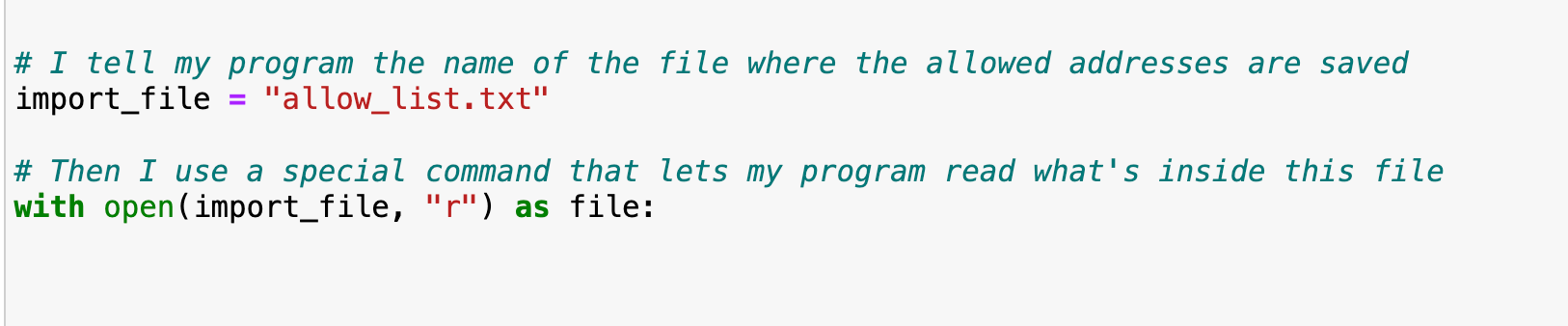
# **Algorithm for file updates in Python**

## **Project description**

At my organization, access to restricted content is controlled with an allow list of IP addresses. The "allow\_list.txt" file identifies these IP addresses. A separate remove list identifies IP addresses that should no longer have access to this content. I created an algorithm to automate updating the "allow\_list.txt" file and remove these IP addresses that should no longer have access.

## **Open the file that contains that allow list**

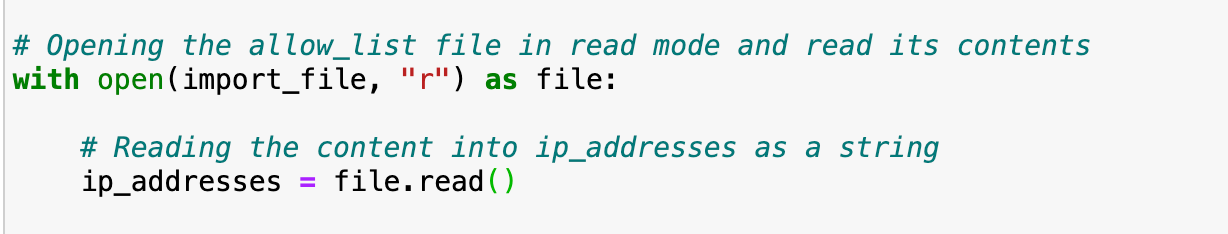
I need to tell the program the name of the file I want to work with. In this case, it's “allow\_list.txt”. I store this name in a variable called import\_file , Then I use a with statement, which is a special way to open files that makes sure they close properly when I’m done with them.



The open() function here is used with two pieces of information the name of the file and the ”r” which stands for “read”. This means it’s just going to look at the file, not change it. The as keyword is followed by file which is going to setting up file as a temporary name for “allow\_list.txt” while it's open.

## **Read the file contents**

To properly handle the file content, the program begins with opening the file in 'read' mode. This mode is signaled by the "r" argument, which tells Python that to only look at the file, not change it.



I applied the .read() method to the file variable, it means I took the action of reading and pointed it at the open file, Then, after capturing all that information in the form of text, I gave it a name, ip\_addresses.

## **Convert the string into a list**

After reading the file, we have all the IP addresses in one big string. To make it easier to work with each IP address, I needed to split this big string into a list. Each item in the list would be one IP address.



Using .split() on string of IP addresses chops it up at every space and gives back a bunch of smaller strings. Each of these smaller strings is an IP address, and they're all stored in a list called ip\_addresses.

## **Iterate through the remove list**

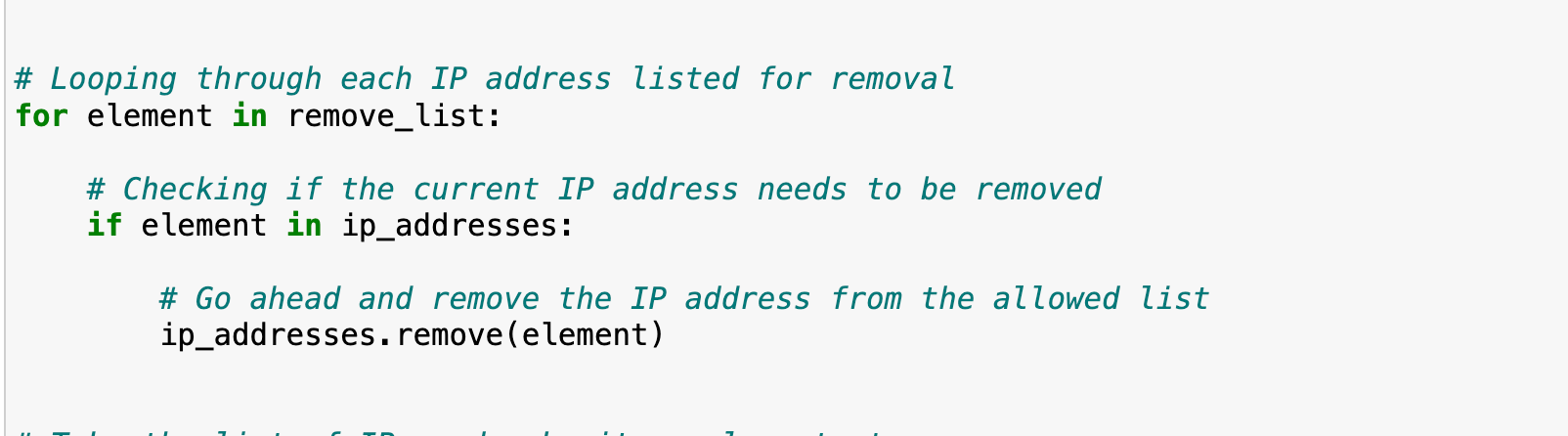
## We have a second list, named remove\_list, which has all the IP addresses we no longer want to allow. To check these against our ip\_addresses list, we need to go through remove\_list one by one and see if any are in ip\_adresses.

## 

## The word for tells Python to start looping. Element is the variable that will hold each IP address from the remove\_list as it go through it.

## **Remove IP addresses that are on the remove list**

To clean up from "allow\_list.txt" by taking out any IP addresses that should no longer be allowed access. These unwanted IP addresses are listed in a separate collection known as remove\_list.



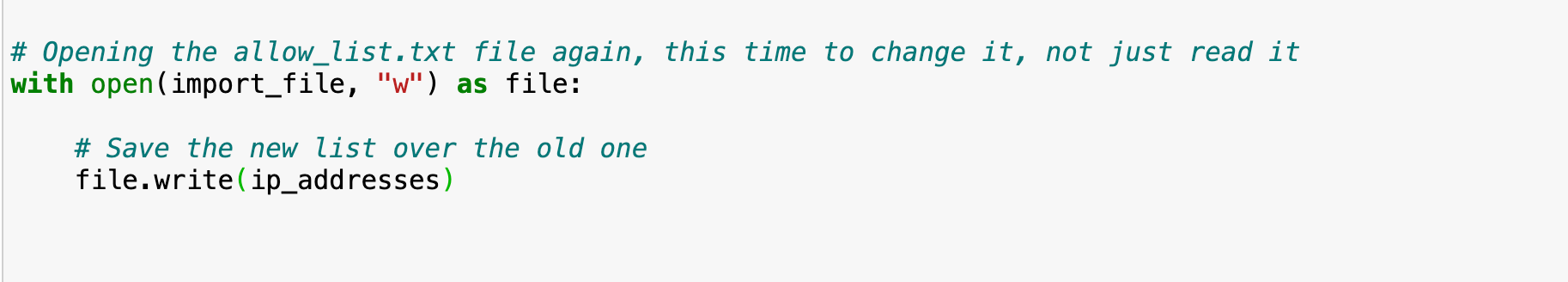
The for loop goes over each entry in remove\_list and The if statement checks if the current IP (element) is in the ip\_address list and If it is, .remove(element) gets rid of it. Each ip that matches gets removed, ensuring that only the IPs we want to keep are left in ip\_addresses.

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

The final task of my algorithm was to save the updated list of IP addresses back into the "allow\_list.txt" file. Before doing that, I needed to turn the list of individual IP addresses back into a single block of text, or a string. The .join() method helped us for this because it takes a list and sticks all the items together with a specified character in between each item. I chose to use a newline character ("\n")which helps us each ip address would end up on its own line in the text file.



Next, I needed to open "allow\_list.txt" again, but this time for writing, which is what "w" stands for in the open()function. This lets me replace the old content with the new, updated list.



By using "w" I told the program that I want to overwrite everything in there with my new list. The .write() function will take that string I just made and printing it onto the file, replacing whatever was there before. So after this, the file only has the IP addresses we want to keep and none of the ones we removed.

## **Summary**

So what I did was I made a Python algorithm effectively maintains the integrity of the "allow\_list.txt" file by removing unauthorized IP addresses that can take a list of Ip addresses we no longer want to have access and remove them from our "allow\_list.txt." This process opens the file, reads the addresses, checks them against the blocked list, takes out the ones we don't want, and then saves the updated list back to the file.