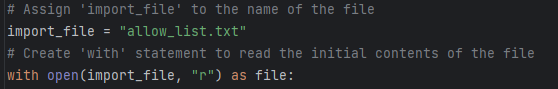
# Algorithm for file updates in Python

## Project description

At my organization, the way we manage access to restricted content is through an approved list of IP addresses stored and regularly updated in a file called "allow\_list.txt". There is also a separate remove list that identifies IP addresses that should not have access to the specified content. I created an algorithm to aid with update automation for the "allow\_list.txt" file and remove any IP addresses that should no longer have access.

## Open the file that contains the allow list

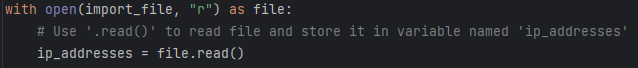
To begin the algorithm, I opened the "allow\_list.txt" file. First, I assigned the file name as a string to the import\_file variable; then, I used a with statement to open the file:



I used the with statement with the .open() function in read mode to open the allow list file to read it. We must do this to access the IP addresses stored inside the file and not the file itself. The with keyword will help manage resources by closing the file after exiting the statement. The “r” value indicates I want to read the file. The code uses the as keyword to assign a variable named file ; it stores the output of the .open() function while working on the statement.

## Read the file contents

To read the contents of the file, we must use the .read() method to turn it into a string.



When using an .open() function how I previously did, we can call the .read() function inside the body of the with statement to turn the file into a string and allows me to read it. I applied the .read() method to the file variable identified in the statement. Then, I stored the output into the variable ip\_addresses to easily reference it later.

## Convert the string into a list

To remove individual IP addresses off the allow list, I need it to convert it to list format from string. To do this, I used the .spli() method to convert the ip\_addresses string to a list:



The .split() function is called by appending it to a string variable. The purpose of turning the string into a list is to make it easier to remove individual IP addresses from the lost. By default, the .split() function accounts for whitespaces as the default splitting element so I left the parenthesis blank. This algorithm will split the input of ip\_addresses into a list, and then store it back into ip\_addresses.

## Iterate through the remove list

This is a key part for this algorithm, as we need to iterate through each IP address that is an element of the remove\_list. To do this, I incorporated a for loop:

A black background with white text

Description automatically generated

The for loop in Python repeats code for a specified sequence. We use this to run a code sequence on every item of the list, which in this case are IP addresses. The for keyword starts the loop, followed by the loop variable element and the keyword in, which instructs the code to iterate through the sequence ip\_addresses and assign each value to the loop variable element.

## Remove IP addresses that are on the remove list

This task involves the removal of certain IP addresses from the allow list ip\_addresses that is also stored in remove\_list. Because there are no duplicates in ip\_addresses, I was able to use the following code:

A screenshot of a computer

Description automatically generated

Firstly, within the for loop, I created a conditional that evaluates if the loop variable element was found in the ip\_addresses list (as using .remove() directly would result in an error). Then, within the conditional, I applied .remove() to ip\_addresses. Passing in the loop variable element as the argument so that each IP address in the remove\_list would be removed from ip\_addresses.

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

As the final step, we need to update the content of the allow list with the revised list of IP addresses. To do this, I first converted the list back into a string:



The .join() method combines all items from a list into a string. In this algorithm, I used .join() to create a string from the list, so that I could pass ip\_addresses in as an argument to the .write() method to write to the file . I used the (“\n”) as the separator to instruct Python to place each item on a new line.

Then, I used another with statement and the .write() method to update the file:

A black background with white text

Description automatically generated

This time, I use the argument of “w” with the function to write to the file. The .write() function writes string data to the file, replacing all previously existing data. I appended the .write() function to the file object that I identified in the statement, and then I passed in the variable as the argument to specify that the contents of the file in the with statement should be replaced with the data in this variable.

## Summary

I created a simple, but efficient algorithm that removes IP addresses identified in a remove\_list variable from a file of approved IP addresses. This algorithm involved opening the file, converting it into a string to be able to read it, then converting this list into a string and storing it in the variable to be able to effectively call and modify the file. I then iterated through each IP address in remove\_list and compared them against each IP address from the “allow\_list.txt” file, verifying for any matches on every element. If there is a match, the .remove() method comes into play to remove the element from ip\_addresses. Finally, I used the .join method to convert the new list into a string, allowing me to rewrite the revised the IP addresses to the “allow\_list.txt” file.