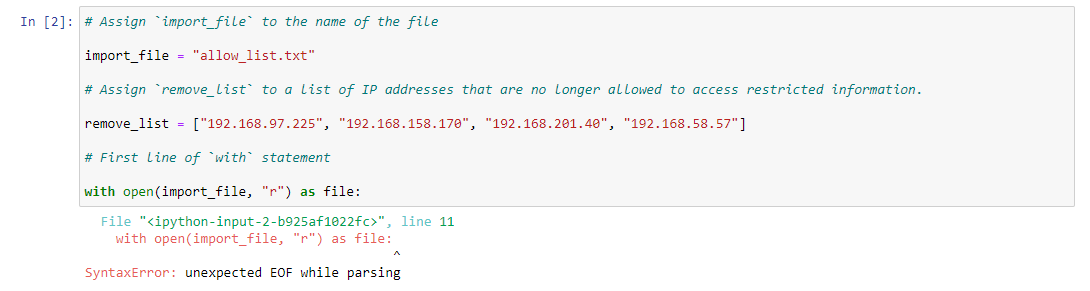
# Algorithm for file updates in Python

## Project description

In this scenario, I am a security professional, who’s job tasks include continuously updating a file identifying employees with access to restricted content for a healthcare company. The content the file refers to is personnel patient records. Access restriction to the contents is based on employee IP addresses.

My task was to use Python code to create an algorithm that would identify any IP addresses on the “Allow” list that are present on the ‘Remove” list, and then remove them from the Allow list.

## Open the file that contains the allow list



The first step is to open the file containing “allow\_list.txt” for the purpose of reading it. This file is assigned to the variable import\_file.

I begin a with statement to open it, using the file variable to store it while I work:

with open(import\_file, “r”) as file:

with begins the statement and the open operator tells Python what to do. In parentheses, the first parameter dictates the txt. file through its assigned variable import\_file, and the second parameter is the operator “r” (telling Python we want to read the file). as references the object we want to store the file in, and file is the object itself. An : is used at the end to complete the statement. Running the code at this point produces the error shown above because the with statement is not yet complete.

## Read the file contents

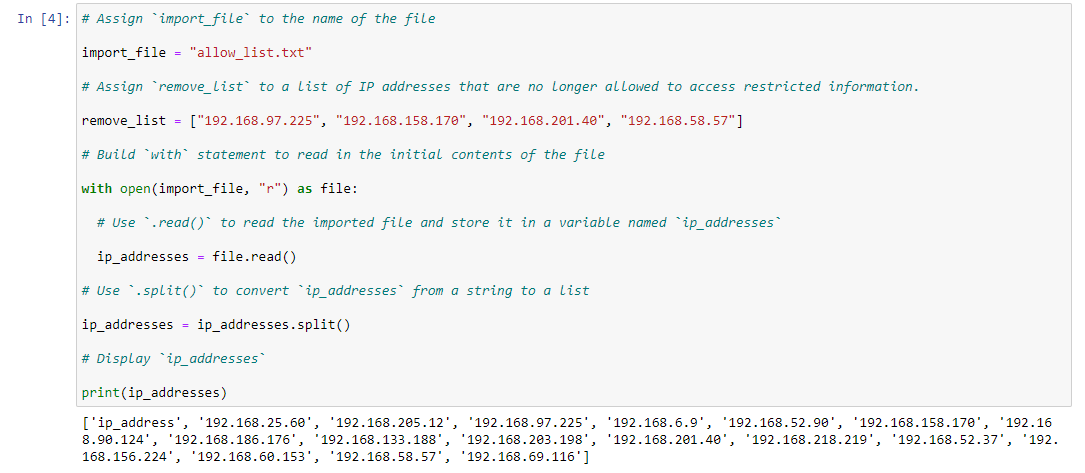


Now that the with statement has been started, I assign the variable ip\_addresses to the read function in order to print the IP addresses contained in the file:

ip\_addresses = file.read()

Because I used file as my object in the with statement. I place it before the .read() function in order to read the imported file. The contents print as a string, one IP address per line.

## Convert the string into a list

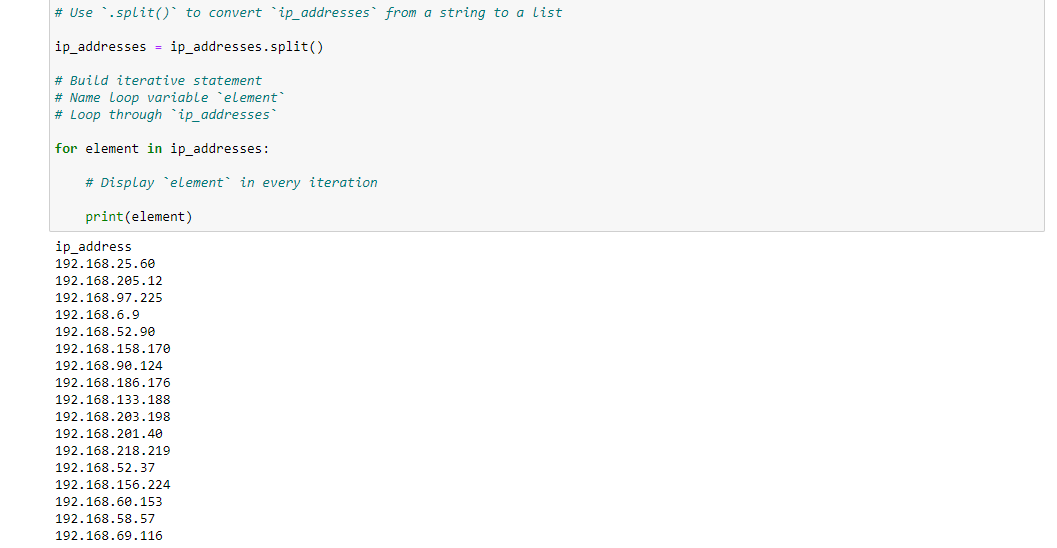


The data needs to be converted from a string into list format so that I can program Python to remove individual IP’s from the allow list:

ip\_addresses = ip\_addresses.split()

I place ip\_addresses before the .split() function to convert the string. When I print(ip\_addesses) now, the data outputs within brackets. By default, the .split() method displays the individual IP’s in quotes, followed by a comma and space.

## Iterate through the remove list

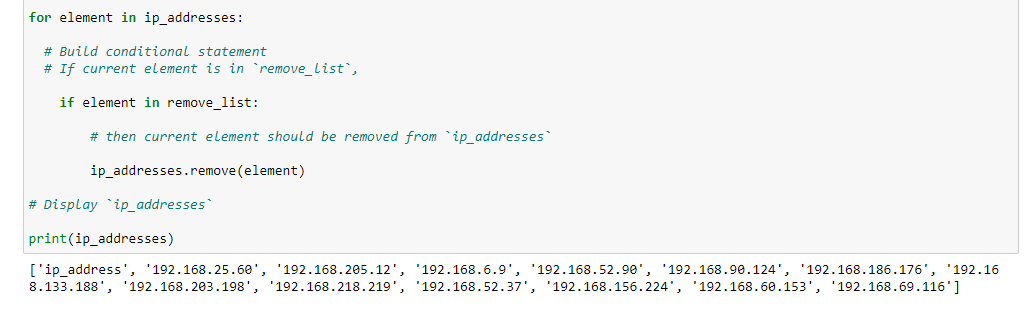


The remove\_list shows the IP addresses that need to be removed. I start by setting up a for loop that will iterate through the elements of the remove list:

for element in ip addresses:

for begins the loop. Following that, I make element our loop variable. in ip\_addresses directs where the iterative statement will loop through. The : is needed at the end as this is the header.

## Remove IP addresses that are on the remove list



Now I add code that checks if IP’s in the Allow list are also in the Remove list, and takes them out if so. I start with a conditional statement:

if element in remove\_list:

This statement begins the body from our header, so it is indented. if begins the conditional, and I apply the element variable for the code to check. in remove\_list dictates where the conditional checks for the variable. This statement also needs to end with a :.

Now I tell Python that if the conditional is met, it will remove the current element:

ip\_addresses.remove(element)

I further indent from the conditional statement above. ip\_addresses is the variable assigned to the list we want to remove from. This is placed before the .remove() function. My element variable is placed within the parentheses so that the remove method applies to the contents in the list.

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



Now that the IP addresses have been removed, I complete the algorithm by updating the file with the revised list. I start with converting ip\_addresses back into a string that can be written into the txt. file:

ip\_ addresses = “ “.join(ip\_addresses)

I apply the .join() function to our ip\_addresses variable with an =. This method converts the list back into a string. “ “ will apply a space in between the IP’s for readability. ip\_addresses is placed within the parenthesis so the join applies to the data.

Finally, I use another with statement to rewrite the original file and call it:

with open(import\_file, “w”) as file:

file.write(ip\_addresses)

For the second parameter in parenthesis, I use the “w” operator for Write. To call the file and rewrite it, I indent and place the object file from the with statement in front of the .write() function with the ip\_addresses variable in parentheses for it to replace.

## Summary

The original text file now contains an updated allow\_list with the IP addresses from the remove\_list removed. The main elements of the algorithm that make it work include the for loop, conditional statement, and .remove() method. The for loop identifies the elements of the list of IP addresses as the data I want Python to loop through. The if statement is the conditional that Python will take action on if the criteria is met (i.e. if the current element from the allow list is also in the remove list). The .remove() function is essential to the algorithm, as this tells Python the specific action to take when the conditional is met (i.e. remove the element from the allow list).