# Algorithm for file updates in Python

## Project description

The goal for this project is to develop an algorithm that completes all of the following:

* Opens and reads a .txt file that contains a list of allowed IP address
* Converts the .txt file from string format to list format
* Iterates through and removes a list of IP Addresses from the .txt file
* Updates and saves the .txt file with the revised list of IP addresses

## Open the file that contains the allow list



To Begin we assign our first variable “import\_file” to the name of the file being imported “allow\_list.txt”. Afterwards we assign our second variable “remove\_list” to a list of IP Addresses that are no longer allowed to access restricted information ‘[“192.168.97.225”, “192.168.158.170”,   
“192.168.201.40”, “192.168.58.57”]’.

To start the algorithm, we begin by opening our file stored in the variable ‘import\_file’ to be read with the command ‘with open(import\_file, “r”) as file:. By beginning with the “with” command, the file being opened will be closed automatically after we’re done with it. To open our file to be read, we use the “open” command, followed by our file “import\_file”, and ending with how the file should be opened (in this case “r” to read the file). After opening the file, we specify under what variable it should be saved to, in this case as file.

## Read the file contents

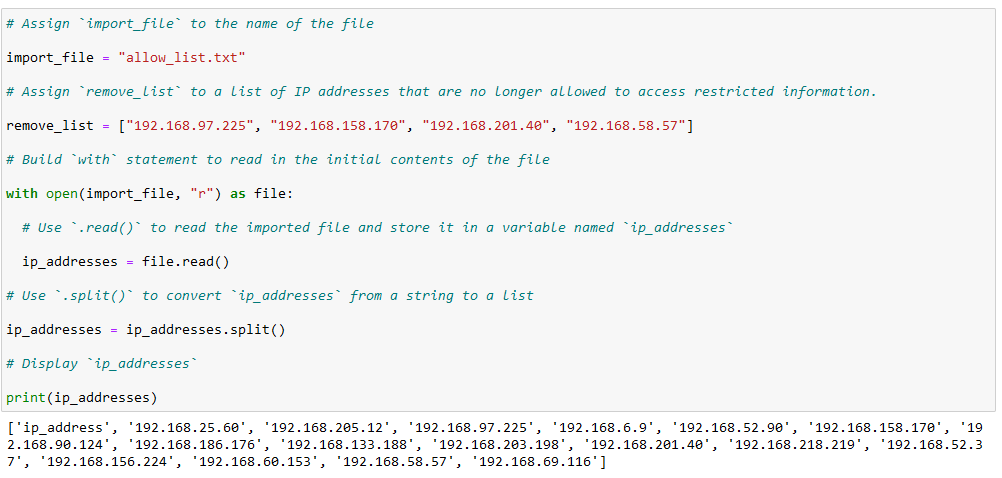


Building off of the previous code, after opening the file we want to read it, store it in a variable, and display it.

To read the file we’ll use the “.read()” command in addition to the variable it should be saved under (in this case “ip\_addresses”). Putting that all together we have the code ip\_addresses = file.read()

In order to display the information stored in the ip\_addresses variable, we’ll use the print command. Putting it all together the command would be print(ip\_addresses).

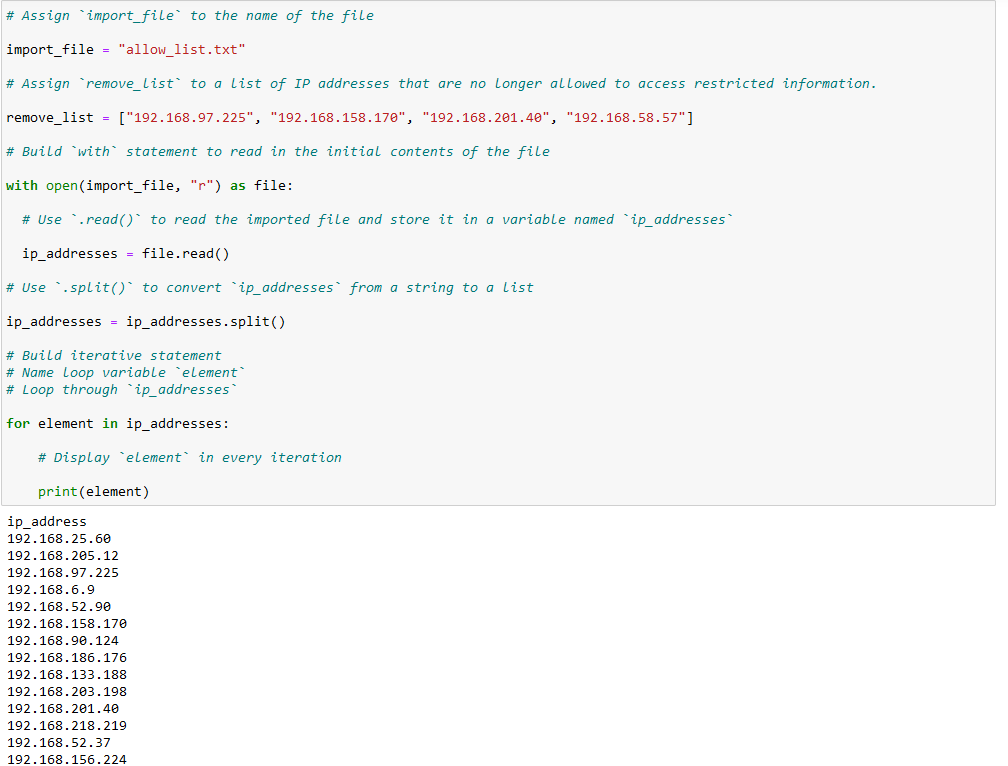
## Convert the string into a list



Typically information that is stored within .txt files is done so in string format, but is much more difficult to decipher within python. In order to present the information in a more user friendly manner, we’ll be using the split command. The split command will convert information stored in string format to list format based off of a specified argument (in our case, a space between text ) Building off of our previous code, in order to convert the string data stored within the ip\_addresses variable to list format, the command will be as such: ip\_addresses = ip\_addreses.split()

By printing the data within ip\_addresses, we can see that the updated list is much more legible than the previous step.

## Iterate through the IP Addresses



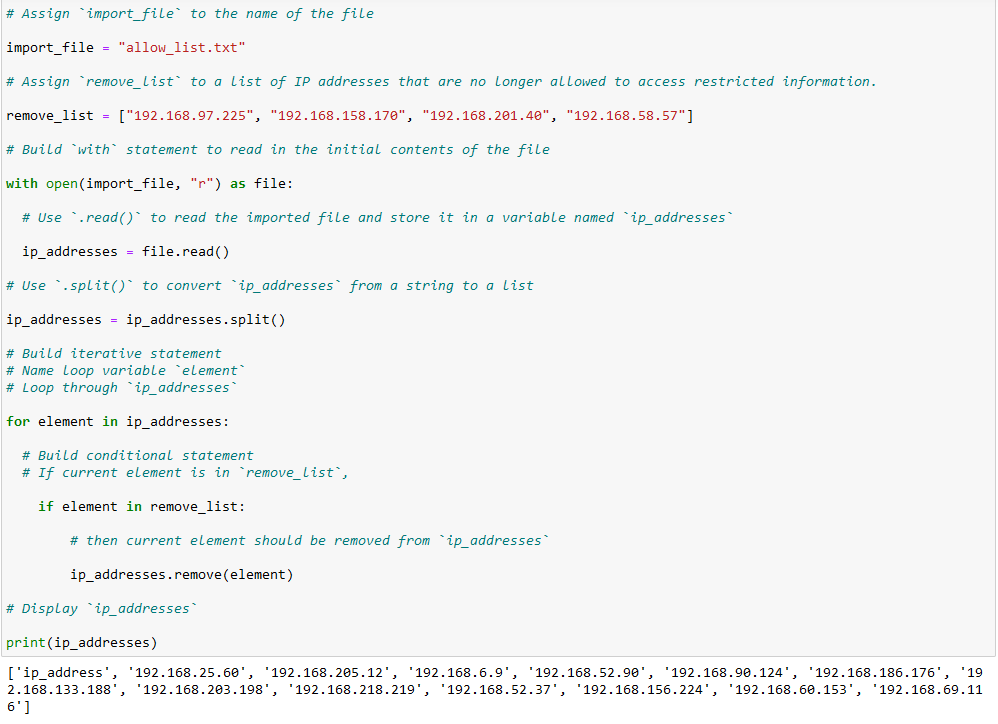
In order to cycle through the IP addresses stored within our variable, we’ll use a for loop. The “for” command will cycle through each item stored within our list, in this case we use the “in” command to tell the script to cycle through each “element” in the ip\_addresses variable.

Putting the command together, it looks like:

**for element in ip\_addresses:**

**print(element)**

## Remove IP addresses that are on the remove list



In the previous step I used a for loop in order to cycle through the elements within the ip\_addresses variable. We’ll use this in conjunction with a conditional statement that will compare the specific element to the elements within the remove\_list variable. If the specific IP is also within the remove\_list, it will be removed from the ip\_addresses variable.

In order to accomplish this, I’ll start with the previous for loop and combine it with a conditional statement using the “if” command. The “if” command compares the following variable to another specified variable, list, etc. In our case, we’re looking to see if “element” is in “remove\_list”. If the specified IP is in the remove list, it will execute the command under it. By using the “.remove” command and specifying “element” as the argument, the IP will be removed from the ip\_addresses variable. Putting it all together, the command looks as such:

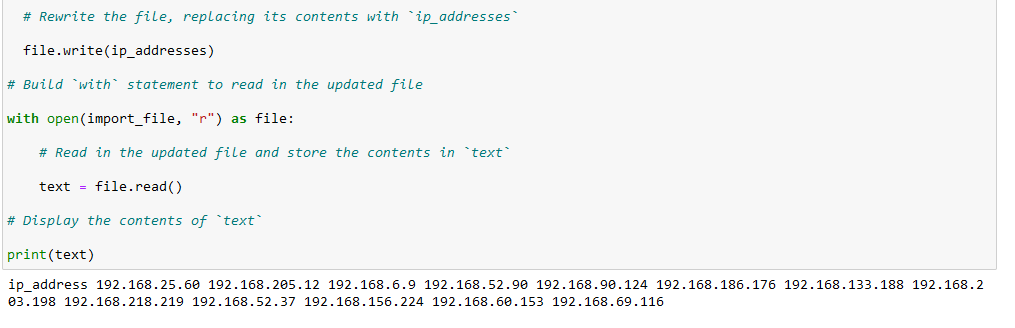
**for element in ip\_addresses:**

**if element in remove\_list:**

**ip\_addresses.remove(element)**

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





Currently the command is set up to run through the list of IP Addresses provided, compare them to the IP Addresses in the remove list, and remove them if they are in both lists. In order to save the updated list, we’ll use the “.join” command in order to adjust the variable from list format to string format, as well as opening and writing over the previous allow list with the updated one. The command will read as such:

**ip\_addresses = “ “.join(ip\_addresses)**

**with open(import\_file, “w”) as file:**

**file.write(ip\_addresses)**

By opening, reading, and printing the current contents of the import\_file variable, it now shows the updated allow list. That command looks like such:

**with open(import\_file, “r”) as file:**

**text = file.read()**

**print(text)**

## Summary

The algorithm created employs a few iterative and conditional statements (in the form of for loops and if statements) in order to easily cycle and filter through the list of given IP addresses. If the IP address is listed on the remove list, it is removed from the list of allowed IP’s, and the list is updated as well. The algorithm also makes good use of opening and writing to .txt files within python as well through the use of the open() command.