# Algorithm for file updates in Python

## Project description

Allow\_list.txt contains a list of IP addresses that are allowed to access restricted information. remove list contains a list of IP addresses that should no longer be in the allow\_list.txt. This algorithm automates updating allow\_list.txt by removing IP addresses that should no longer be in it.

## Open the file that contains the allow list

This assigns the file name allow\_list.txt to the variable import file.

# Assign `import\_file` to the name of the file

This opens the file.

import\_file = “allow\_list.txt”

The with statement opens the file for reading and closes it at the end of the with statement. As stores the contents of import\_file in the variable file for use within the with statement.

# First line of `with` statement

with open(import\_file, "r") as file:

## Read the file contents

As above.

with open(import\_file, “r”) as file:

This will convert the contents of file into a string and assign the results to the variable ip\_addresses for use in the with statement.

# Use `.read()` to read the imported file and store it in a variable named `ip\_addresses`

ip\_addresses = file.read()

## Convert the string into a list

This will convert the string data returned from ip\_addresses into a list and assign the results back to the variable ip\_addresses. The split is delineated by whitespaces.

# Use `.split()` to convert `ip\_addresses` from a string to a list

ip\_addresses = ip\_addresses.split()

## Iterate through the remove list

This begins the loop. The for statement is used to apply specific statements to all element in the sequence.

# Build iterative statement

# Name loop variable `element`

# Loop through `remove\_list`

for element in remove\_list:

# Display `element` in every iteration

print(element)

## Remove IP addresses that are on the remove list

The algorithm will need to remove an IP address in remove\_list from ip\_addresses. There were no duplicate IP addresses in this example so .remove() could be used. First, the contents of remove\_list were assigned to the variable element. Next, an if statement was used to search ip\_addresses for element. If an IP address in remove\_list was found, it was deleted from ip\_addresses.

for element in remove\_list:

# Create conditional statement to evaluate if `element` is in `ip\_addresses`

if element in ip\_addresses:

# use the `.remove()` method to remove

# elements from `ip\_addresses`

ip\_addresses.remove(element)

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

.join is used to create a string to be used as an argument in .write(). “\n” is used as the separator to tell Python to place each element on a new line.

# Convert `ip\_addresses` back to a string so that it can be written into the text file

ip\_addresses = "\n".join(ip\_addresses)

Using with, the file contained in the variable import\_file was opened for writing and assigned to the variable file for use within the with statement.

# Build `with` statement to rewrite the original file

with open(import\_file, "w") as file:

This will write the string data to the file contained in the variable ip\_address and will overwrite the existing contents. The end result is that the IP addresses in the remove\_list are deleted from the allow\_list.txt file.

# Rewrite the file, replacing its contents with `ip\_addresses`

file.write(ip\_addresses)

## Summary

This algorithm was created to automate maintenance on the file allow\_list.txt by removing IP addresses in the remove\_list. Allow\_list.txt contains a list of IP addresses that are allowed to access restricted information. First, the allow\_list.txt file was opened and the contents were stored as string data. Next the string data was converted to a list. Afterward the list data was iterated through and entries from the remove\_list were deleted for ip\_addresses. Finally, the resulting data was converted back into a string and written into the file allow\_list.txt with the updated list of allowed IP addresses.