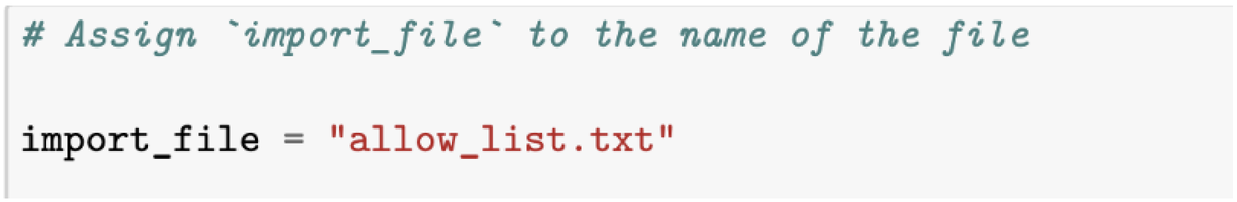
Algorithm for file updates in Python

Project description

A very important aspect of cybersecurity is controlling access to restricted content. Our job as cybersecurity analyst is to work with files that contain IP addresses that are allowed specific restricted content at our organization. I have created an algorithm with Python to automate the process of parsing files keeping them up to date as well as removing IP addresses that no longer have access to the restricted content.

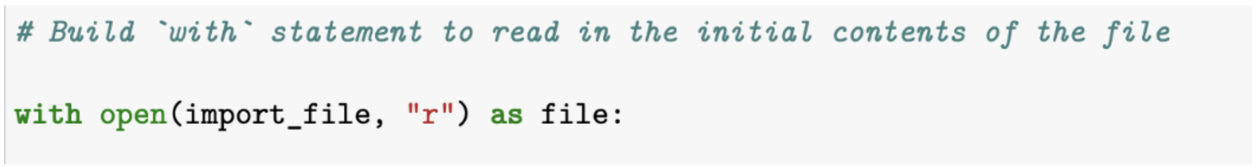
Open the file that contains the allow list

To open the file that contains the allow list, we must first create a part of the algorithm.



```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"
```

Then we use a **with** statement to open it. We will use the variable **file** to store the file while we work on it with the inside statement.

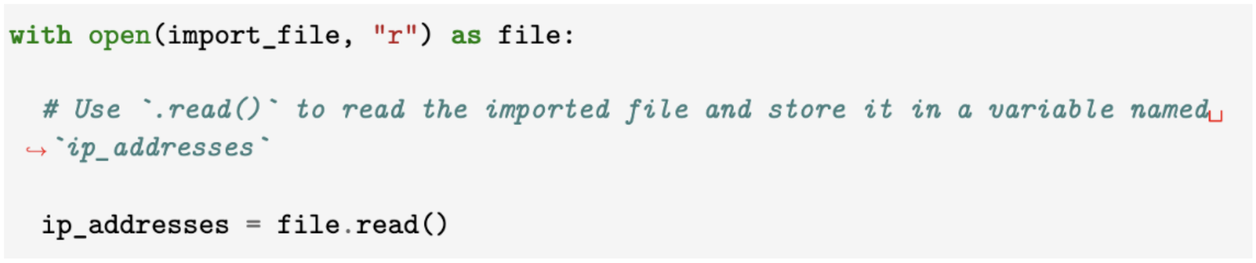


```
# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:
```

In the algorithm our with statement is used with the open() function because it allows the algorithm to open the important file and read it back to us as the output. The reason why we do this is because we want to get a list of IP addresses stored within.

Read the file contents

When reading the contents within the file, we must use a specific method to do so. We use the with open function to open the file, use the .read() method to convert the contents of the allow list file into a string so we can read them. We will call this variable ip_addresses. The with statement is used to ensure that it is properly closed after reading.

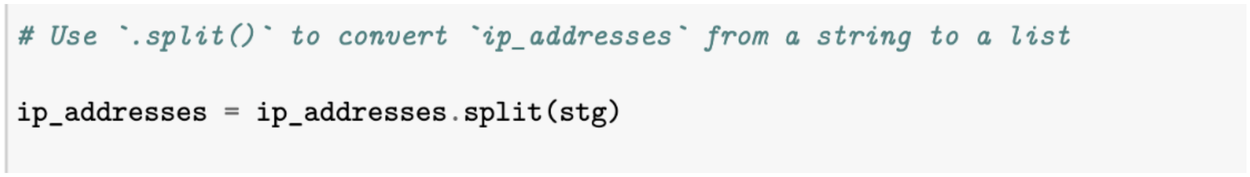


```
with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named
↪`ip_addresses`

  ip_addresses = file.read()
```

Following that, I assigned the output of the file reading to ip_addresses and the code reads the contents in the allow list to extract the data needed.

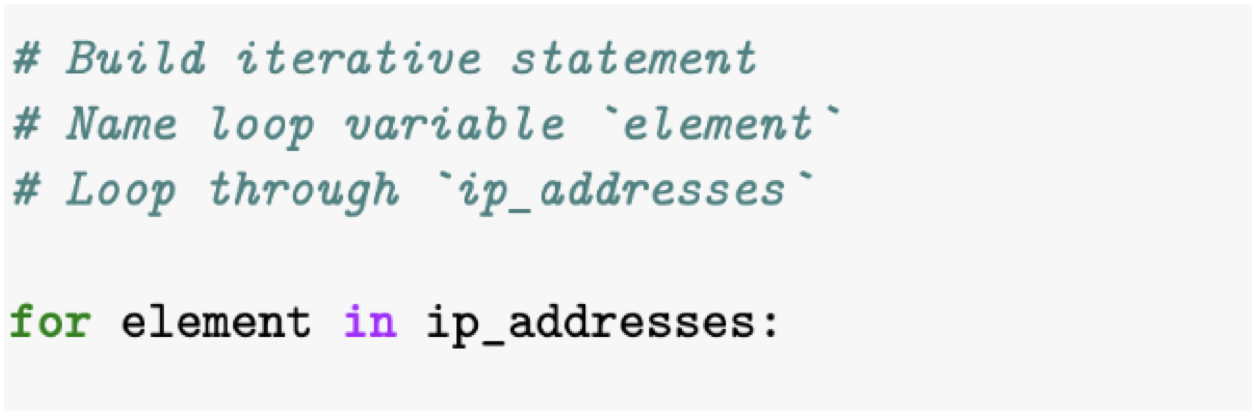
Convert the string into a list

In order to remove the individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, I used the .split() method to convert the ip_addresses string into a list. Here below I got a bit carried away and typed in the stg function but ultimately it was not needed. The reason why is because it makes it easier to to remove from the allow list and therefore helps the automation process speed up. (Each string is spaced out)



```
# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split(stg)
```

Iterate through the remove list



```
# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:
```

Furthermore, I then create a second list called the remove list that contains all of the IP addresses that should be removed from the ip_addresses list. To do this I set up a header of a for loop that will iterate through the remove_list. I then use the element as our loop variable to assign each value.

Remove IP addresses that are on the remove list

As I continue removing IP addresses that are on the remove list, in the body of the iterative statement I add code that will do so. First, I create a conditional that evaluates if the loop variable element is part of the ip_addresses list. Secondly, within that conditional, I apply the .remove() method to the ip_addresses list. Lastly, within that conditional I apply the .remove() method to the ip_addresses list and remove the IP addresses identified in the loop variable element.

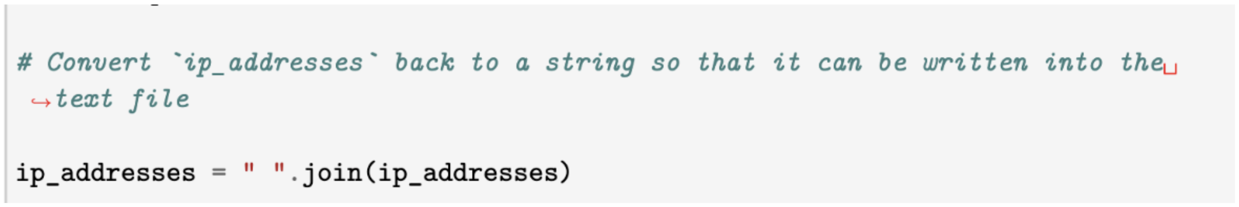
```
for element in ip_addresses:
```

```
  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)
```

In addition, the .remove() method is used to delete a specific element from the ip_addresses list. This approach works because there are no duplicate values in the list, ensuring that .remove() only removes the intended IP address without affecting other elements.

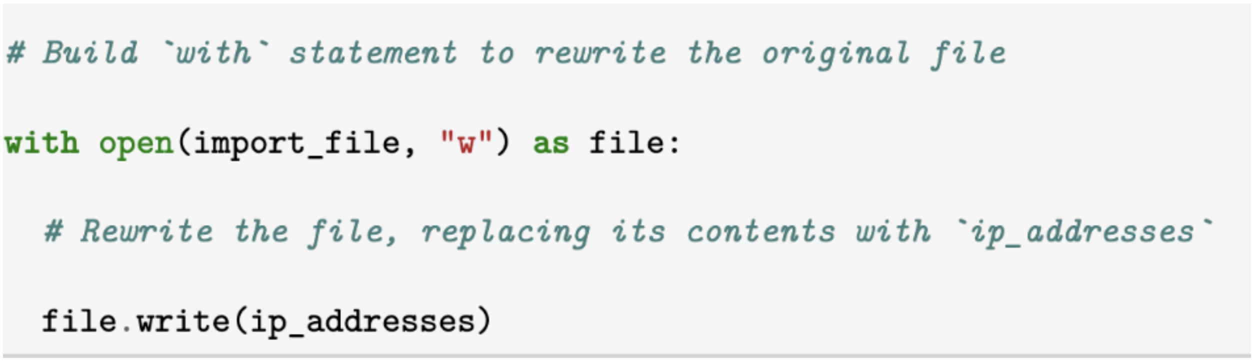
Update the file with the revised list of IP addresses

Now that I have removed the Ip addresses from the ip_address variable, I can complete the algorithm by updating the file with this revised list. In order to do so I first convert the ip_addresses list back into a string using the .join() method and apply the method to the string “\n” in order to separate the elements in the file by placing them on a new line.



```
# Convert `ip_addresses` back to a string so that it can be written into the␣
 ↪text file

ip_addresses = " ".join(ip_addresses)
```

The .join() method combines all items into a string and with this algorithm I use the method to create a strong from the list of ip_addresses so that I could pass it as an argument to the .write() method when writing it to the allow list. I then used the string once again “\n” to separate and instruct the algorithm to start each output as a new line.



```
# Build `with` statement to rewrite the original file

with open(import_file, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

Furthermore, now I used a second argument of “w” with the open() function. This lets the algorithm know that I want to open a new file and write over the contents/replace content. I then write it as a .write() function in the body of the with statement. By doing so the restricted content will no longer be accessible to any IP addresses that were removed from the allow list. This allows me to rewrite the following file within the statement and pass it in the ip_addresses variable as the argument that it can specify to the contents of the statement. The restricted content will now no longer be accessible to any IP addresses that were removed from the willow list and to rewrite the file, I appended the .write() function to the file that I identified in the with statement

Summary

I created an algorithm with Python that removes IP addresses identified in the remove_list from the allow_list.txt file of approved IP addresses. The python algorithm involved opening the file, converting it into a string to be read, then converting the string into a list (ip_addresses), I then iterated through the IP addresses through the remove list. Each time I had to make sure the element was part of the ip_addresses list and if it was , I would use the .remove() method to remove it from the following. Following that, I used the .join() method to convert the ip_addresses back into string so that I could write them into the allow_list file with the new and updated algorithm of revised IP addresses.