Algorithm for file updates in Python

Project description

I am a security professional working at a healthcare company. Access to restricted content is controlled through an allow list file called “allow_list.txt” which contains valid IP addresses. We also have a remove list called “remove_list.txt” which contains IP addresses that need to be removed from the allow list. I created an algorithm that automatically updates the allow list based on the contents of the remove list.

Open the file that contains the allow list

I started by opening the “allow_list.txt” file and assigning it as a string to a variable called `import_file`:

```
import_file = "allow_list.txt"
```

I then opened the file using a `with` statement:

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

Using `with` and `open()` together allows a file to be opened in the python environment. The `open()` function contains two arguments, one for the file that you want to open and the second for what you need to do to the file. In this case we are reading the file so I entered “r” as the argument. I also used the `as` function to assign a variable name to the file. Here I simply assigned it to the variable “file”.

Read the file contents

To read file contents, I used the `.read()` function which converts it into a string.

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

    ip_addresses = file.read()
```

After opening the file with the previous functions, the .read() function can be used in the body of the with statement. The .read() function is applied to the file variable, the output of which is then assigned to the ip_addresses variable for future use.

Convert the string into a list

In order for IP addresses to be removed from the allow list, the data needs to be in a list format. By using the .split() function, I can convert the current content of the ip_addresses variable into a list because it is currently a string:

```
ip_addresses = ip_addresses.split()
```

By default the .split() function automatically splits string data by whitespace so no parameters are needed for this particular file. I then reassigned the output of this function to the same ip_addresses variable.

Iterate through the IP addresses list

This part of the algorithm involves iterating through the ip_addresses list using a for loop:

```
for element in ip_addresses:
```

For loops are used to repeat code in a specified sequence. In this example, every item in the ip_addresses list will be temporarily assigned as an element that will be used in future code to check if the specific item needs to be removed.

Remove IP addresses that are on the remove list

Any IP address in the ip_addresses list that is also in the remove_list must be removed. This can be done using the following code:

```
for element in ip_addresses:
    if element in remove_list:
        ip_addresses.remove(element)
```

After the for loop is initiated, any element in the remove_list gets removed from the ip_addresses list. This is done using an if statement and the in keyword. If the element is found

to be in the remove_list, I use the .remove() function and pass element as the argument. This removes the matching string from the ip_addresses list.

Update the file with the revised list of IP addresses

As the final step, the allow list file needs to be updated with the new list of addresses. The list first needs to be converted back to a string, as this is done using the .join() function:

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

The .join() function combines all elements of a list into a string. You need to specify what character should be placed in between each list item. Because we originally split the list by whitespace, we will combine the list using whitespace. This is represented by the " " before the .join() function.

Next, I updated the file using another with statement and the .write() function:

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

    file.write(ip_addresses)
```

By using the "w" argument, I can overwrite the previous version of the file which contains the incorrect IP addresses. In the body of the with statement I used the .write() function on the file variable and passed it the ip_addresses list as an argument which overwrites the file.

Summary

This algorithm can be used to remove IP addresses from the "allow_list.txt" by removing all addresses found in the remove_list. The algorithm starts by opening the "allow_list.txt" file, converting it to a string, and then converting the string into a list. This list is then stored in the ip_addresses variable. Next, using a for loop I iterated through the remove_list and evaluated if each element was in ip_addresses. When it was, I used the .remove() function to remove the element from ip_addresses. I then converted ip_addresses back into a string and saved the new list by overwriting the contents of the "allow_list.txt" file.