Algorithm for file updates in Python

Project description

This little project consists in developing an algorithm to manage ip addresses from a list of permissions into a restricted subnetwork using the programming language Python.

Open the file that contains the allow list

Assigning the file name/path to a variable thus allows code reusability.

```
Import_file = "allow_list.txt"
```

Using 'with' statement as exception handling execute the opening process
'open' function allows to open a file
'r' statement indices that we are opening the file in 'reading' mode
'as' works as alias and allows to assign a statement or return to a variable
'file' is the variable assigned with the result of the file opening process

```
with open(import_file, “r”) as file:
```

Read the file contents

Converting the contents of the file into a string variable using the .read() method

```
ip_addresses = file.read()
```

Convert the string into a list

Converting the contents of the string into a list variable using the .read() method allows removing individual IP addresses.

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

Iterate through the remove list

We need to iterate the ip_addresses_list to remove all the IP addresses in the remove_list.
Step 1: Create a for loop to iterate the remove_list;
#'for' indicates the beginning of a loop
#'element' is the loop variable
#'in remove_list' indicates the iterable object to use

```
for element in remove_list:
```

Remove IP addresses that are on the remove list

Step 2: Verify the existence of the element in the ip_addresses_list with ‘if’ condition to avoid program’s issues;
Step 3: Use .remove() method to remove the element if the condition from strep 2 confirms;

#’.index()’ returns the index of the first occurrence of an element. To be certain the element exists in the list, the index should return a number >=0;
#’.remove()’ Removes the first occurrence of an element in a list, we use this method to remove the IP address present in both lists.

```
        if ip_addresses_list.index(element) >= 0:
                ip_addresses_list.remove(element)
```

Update the file with the revised list of IP addresses

Now we pretend to update the previous read file without the removed IP addresses.
Step 1: Convert again the ip_addresses list back into a string and separating elements in a new line;
Step 2: Open again the file and overwrite with the new data;

#’.join’ method allows us to convert a list into a string and the ‘\n’ indicates the element’s separator.

```
ip_addresses = “\n”.join(ip_addresses)
```

#Applying again the open method in ‘writing’’ mode to overwrites the file
#’.write’ method allows us to overwrite the file with the new data or create a new file if it doesn't exist.

```
With open(import_file, “w”) as file:
        file.write(ip_addresses)
```

Summary

With this small project, we applied knowledge related to file handling, string and list manipulation, conversions, loops, and conditions. These topics are essential to comprehend as they are frequently used in the realm of cybersecurity, enabling us to manage accesses, users, and various scenarios that depend on reading and comparing files.

In this project, I began by creating a global variable to hold the name of the target file. Utilizing the 'with' statement and the 'open' function, I opened the file in 'read' mode. Subsequently, I read its contents into a string variable and converted it into a list, making it iterable. The objective was to compare this list with another list containing addresses to be removed, and then eliminate the matching addresses.

To achieve this, I employed a loop to iterate through the list of addresses to be removed. For each element, I checked whether it existed in the access list. If the defined condition was met, I proceeded to remove that element. Lastly, I reopened the file in 'write' mode and updated all the data.