Algorithm for file updates in Python

Project description

At my organization, access to restricted content is controlled with an allow list of IP addresses. The `"allow_list.txt"` file identifies these IP addresses. A separate remove list identifies IP addresses that should no longer have access to this content. I created an algorithm to automate updating the `"allow_list.txt"` file and remove these IP addresses that should no longer have access.

Open the file that contains the allow list

Here I opened the allow_list.txt file as 'import_file' and proceeded to open it with a 'with' statement.

The purpose of opening the file is to view the ip addresses that are in the allow list. In my algorithm, the 'with' statement opens the allow list file in read mode so I can access the stored IP addresses. Using 'with open(import_file, 'r') as file:' ensures the file is properly closed after use, while 'r' specifies read mode and 'file' serves as the temporary variable holding the file's contents.

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# First line of `with` statement

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

Read the file contents

To read the file contents, I used the '.read()' method to convert it into a string.

Using 'with open(import_file, 'r') as file:' , I applied the '.read()' method to convert the file contents into a string and stored it in the variable 'ip_addresses'. This allowed me to access the data from 'allow_list.txt' in a format I could later organize and extract within my Python program.

```
with open(import_file, 'r') as file:
    ip_addresses = file.read()
```

Convert the string into a list

To remove individual IP addresses from the allow list, it needed to be in a list format. To convert 'ip_addresses' into a list, I used the '.split()' method. This made it easier to manage and later make it simpler to remove the specific addresses from the allow list that needed to be removed since each ip became its own element.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

with open(import_file, "r") as file:
    ip_addresses = file.read()

ip_addresses = ip_addresses.split()

print(ip_addresses)
```

Iterate through the remove list

To be able to remove the given ip addresses from the allow list, I needed to create a ‘for’ loop to simplify the process to automatically apply the specified code to all elements.

```
for ip in remove_list:
```

Remove IP addresses that are on the remove list

Now that my ‘for loop’ was created, I needed to create the processes which would remove the elements from the allow list. Since there are no duplicates, the following code was able to be used to remove each ip from the remove list from the allow list.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

with open(import_file, "r") as file:
    ip_addresses = file.read()

ip_addresses = ip_addresses.split()

for ip in remove_list:
    if ip in ip_addresses:
        ip_addresses.remove(ip)

print(ip_addresses)
```

Update the file with the revised list of IP addresses

As the final step, I needed to now update the allow list file with the revised list that had now removed the ‘remove_list’ ip’s. First the list needed to become a string again, ‘.join()’ was able to do this. Then, using a ‘with’ statement and the ‘.write()’ method, I updated the file.

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

ip_addresses = "\n".join(ip_addresses)


with open(import_file, 'w') as file:
    file.write(ip_addresses)
```