Algorithm for file updates in Python

Project description

You are a security professional working at a health care company. As part of your job, you're required to regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

Your task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, you should remove those IP addresses from the file containing the allow list.

Open the file that contains the allow list

```
import_file = "allow_list.txt"

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

Shows the variable Import_file being used to hold the allow_list.text. The with statement was then used to .open() the variable Import_file with the read parameter, this open instance of the file with the read parameter was opened as “file”.

Read the file contents

```
import_file = "allow_list.txt"

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

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.
168.186.176 192.168.133.188 192.168.218.219 192.168.52.37 192.168.156.2
24 192.168.60.153 192.168.69.116
```

The .read() function was used to make the file readable in the python code. It was then printed to screen using the print() function as a test and to examine the layout of the imported file.

Convert the string into a list

```
import_file = "allow_list.txt"

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

```
['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.1
68.90.124', '192.168.186.176', '192.168.133.188', '192.168.218.219', '1
92.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

As the file was one long string, the .split() function was used to make the string into a list for easier parsing and editing.

Iterate through the remove list

```
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "1

with open(import_file, "r") as file:
    ip_addresses = file.read()
    ip_addresses = ip_addresses.split()
    print(ip_addresses)
for element in remove_list:
    print(element)
```

```
['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.1
68.90.124', '192.168.186.176', '192.168.133.188', '192.168.218.219', '1
92.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
192.168.97.225
192.168.158.170
192.168.201.40
192.168.58.57
```

A for loop was used to cycle through each element in the remove_list. The remove list element was then printed to screen to ensure that it functioned properly.

Remove IP addresses that are on the remove list

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

The for loop was then edited with an if statement so if the element pulled from the remove_list and stored as an element was found in the ip_addresses list, it would use the .remove() function to remove the matching element from the ip_addresses list.

Update the file with the revised list of IP addresses

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

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

The ip_addresses variable was then updated using the .join() function then written back onto the import_file via the with open() function with the write parameter.

Summary

Overall, this algorithm uses Python code to check whether the allow list file contains any IP addresses identified on the remove list. If any IP addresses on the remove list are found on the current imported file of IP addresses, it will remove that IP address from the imported file.