Algorithm for file updates in Python

Project description

As a security professional, I'm expected to regularly update a file that identifies employees who can access restricted content. My 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, I should remove those IP addresses from the file containing the allow list.

Open the file that contains the allow list

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

import_file = "allow_list.txt"

# First line of `with` statement

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

The file `"allow_list.txt"` is assigned to the variable `"import_file"`. Then, I used a `with` statement to open the file. The first parameter inside the `"open()"` function has the location of the file which is stored in the variable `"import_file"`. The second parameter of the `"open()"` function for now is `"r"` which indicates to read the file. Then, the variable `file` is used to store the file while I work with it inside the `with` statement.

Read the file contents

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

    ip_addresses = file.read()
```

To read a file, it must be converted to a string first by using the `.read()` method. The `file` variable is used to generate a string of the file contents through `.read()`. This string is then stored in another variable called `ip_addresses`.

Convert the string into a list

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

    ip_addresses = file.read()

ip_addresses = ip_addresses.split()
```

I used the `.split()` method to convert the `ip_addresses` string into a list, and reassign the variable `ip_addresses` with this new data type. This is necessary in order to remove individual IP addresses from the allow list.

Iterate through the remove list

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

```
for element in remove_list:
```

A second list called `remove_list` contains all of the IP addresses that should be removed from the `ip_addresses` list. First, I built an iterative statement by setting up a `for` loop. Then, I named a loop variable `element` which will loop though `remove_list`.

Remove IP addresses that are on the remove list

```
for element in remove_list:

    if element in ip_addresses:

        ip_addresses.remove(element)
```

A conditional statement is created to evaluate if the loop variable `element` is part of the `ip_addresses` list. Then, within the conditional, the `.remove()` method is applied to the `ip_addresses` list and removes the IP addresses identified in the loop variable `element`.

Update the file with the revised list of IP addresses

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

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

After removing the IP addresses from the `ip_addresses` variable, the file can be updated with the new revised list. First, the `ip_addresses` must be converted back into a string by using the `.join()` method. The `.join()` is applied to the string `"\n"` in order to separate the elements in the file by placing them on a new line. Then, I used another `with` statement and the `.write()` method to write over the file assigned to the `import_file` variable.

Summary

I managed to open and read the file by using a `with` statement and the `.read()` method. Then, I proceeded to convert the IP addresses into a list by using the `split()` method in order to remove them individually. Then, I set up a `for` loop that will iterate through the `remove_list`. Inside this loop, I added a conditional statement which evaluates if the loop variable is part of the `ip_addresses` list, and if so, the `.remove()` method will be applied to that list. Finally, the `ip_addresses` list is converted back into a string by using the `.join()` method, and to update it, I used a `with` statement and the `.write()` method to write over the file assigned to the `import_file` variable.