Algorithm for file updates in Python

Project description

As a security professional in a health care organization, I needed a way to efficiently update a file that lists which employees by, IP address, can access restricted content. The goal was to develop a Python algorithm that checks the allow list for IP address found on a separate remove list and removes them if necessary. Automating this task helps keep the access list up-to-date and secure, making sure only authorized employees are allowed access.

Open the file that contains the allow list

To access the allow list, I used Pythons `with` statement and the `open( )` function. The `with` statement is important because it handles closing the file automatically, even if an error occurs. The `open ( )` function takes two arguments: the filename and the mode. Here, I used `"r"` to open the file in read mode.

```
import_file = "allow_list.txt"
with open(import_file, "r") as file:
      # file is now open for reading
```

Read the file contents

To read the content of the file, I used the `.read( )` method on the `file` object, which returns the file's contents as a single string.

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

Convert the string into a list

Since the IP addresses are stored as a single string, I used the `.split( )` method to turn that string into a list. This allows me to iterate over each IP address individually.

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

Iterate through the remove list

I then looped through each element in the `remove_list` using a **for** loop. This makes it easy to check each IP address that needs to be removed.

```
for element in remove_list:
      # check if element is in ip_addresses
```

Remove IP addresses that are on the remove list

Inside the loop, I used a conditional statement to see if the current IP address (`element`) exists in the allow list (`ip_addresses`). If it does not, I removed it with the `.remove( )` methos. This method works here because we know the allow list has no duplicates.

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

Update the file with the revised list of IP addresses

After updating the list, I used the `.join()` method to convert the list back into a string. I separated each IP address by a newline character (`"\n"`), so each address appears on a new line in the file. Then, I opened the file again in write mode (`"w"`) and wrote the updated string back to the file using the `.write()` method.

```
ip_addresses = "\n".join(ip_addresses)
with open(import_file, "w") as file:
      file.write(ip_addresses)
```

Summary

This algorithm streamlines the process of updating an allow list for restricted content by:

- Opening the file containing IP addresses.
- Reading its contents and converting them into a list.
- Iterating through a separate remove list and removing any matching IP addresses from the allow list.
- Writing the updated list back to the file, ensuring the allow list always reflects current access permissions.
  Using Python for this process makes the task efficient, repeatable, and less prone to human error. The logic can easily be adapted for any similar file update or access-control task in a cybersecurity setting.