Algorithm for file updates in Python

Project description

In this scenario, I am working as a security professional at a healthcare company responsible for maintaining access control to a restricted network. Employees are granted or removed from access based on their IP addresses, which are stored in an allow list. A separate remove list contains IP addresses that must be taken out of the allow list due to changes in employee roles or access permissions. My task is to use Python to automate the process of checking the allow list and removing any IPs that appear on the remove list.

Open the file that contains the allow list

To begin, I assign the filename `"allow_list.txt"` to a variable called `import_file`. Then I use a `with` statement to open the file in read mode and store the file object in the variable `file`. The `with` statement automatically closes the file when the block is finished.

```
import_file = "allow_list.txt"

with open(import_file, "r") as file:
   # work with the file inside this block
   ip_addresses = file.read()
```

Read the file contents

Inside the `with` block, I use the `.read()` method to read the entire contents of the file and store them as a single string in the `ip_addresses` variable. This gives me one string that contains all of the IP addresses from the allow list.

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

Convert the string into a list

Because I need to remove specific IP addresses, I convert the ip_addresses string into a list so that each IP address is a separate element. I use the .split() method, which breaks the string at whitespace or newline characters.

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

Iterate through the remove list

The program uses a second list called `remove_list` that contains the IP addresses to be removed from the allow list. I set up a `for` loop that goes through each element in `remove_list` using the loop variable `element`.

```
remove_list = ["10.0.0.5", "10.0.0.9"]  # example IPs to remove

for element in remove_list:
   # removal logic goes here
   ...
```

Remove IP addresses that are on the remove list

Inside the loop, I check whether the current `element` exists in the `ip_addresses` list. If it does, I call the `.remove()` method to delete that IP address from the allow list. This works because there are no duplicate IP addresses in the list, so each call to `.remove()` only removes one specific entry.

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

Update the file with the revised list of IP addresses

After all of the unwanted IPs have been removed, I convert the updated list back into a string so it can be written to the file. I use the "\n".join(ip_addresses) expression to place each IP address on its own line. Then I open the same file again in write mode ("w") and use .write() to overwrite it with the cleaned list of IP addresses.

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

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

Summary

This algorithm automates the process of managing an IP allow list by reading the file contents, converting them into a list, and comparing them against a separate remove list. It uses a `for` loop and conditional logic to find matching IP addresses and the `.remove()` method to delete them from the allow list. After the removals, it rebuilds the list into a newline-separated string and writes it back to the original file using `.write()`. By automating these steps, the script helps keep network access controls accurate and reduces the risk of human error when updating security files.