Algorithm for file updates in Python

Project description

Managing access to restricted content often relies on an allow list of approved IP addresses. In this case, the approved IPs were stored in a file called `allow_list.txt`, while a separate `remove_list` held IPs that should be denied access moving forward. To streamline the process and reduce the chance of human error, a Python script was created to automatically update the allow list by removing any IPs found in the `remove_list`.

Open the file that contains the allow list

The script begins by assigning the name of the allow list file to a variable:

```
import_file = "allow_list.txt"
```

To safely open and read the file, a `with` statement is used:

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

Using `with` ensures the file is closed automatically after reading, which is both cleaner and safer when handling files.

Read the file contents

The contents of the file are read and stored as a string:

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

This converts all listed IP addresses into a single string, which can then be manipulated as needed.

Convert the string into a list

In order to make individual IPs easier to work with, the `.split()` method is used:

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

Since the IPs in the file are separated by whitespace, `.split()` breaks them apart and stores them as individual elements in a list called `ip_addresses`.

Iterate through the remove list

To begin removing access, a `for` loop is used to examine each IP in the `remove_list`:

```
for element in remove_list:
```

This loop goes through the list one IP at a time, preparing for a conditional check to determine whether each IP exists in the allow list

Remove IP addresses that are on the remove list

A conditional inside the loop checks whether the current IP is in the allow list before attempting to remove it:

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

Using an `if` statement here prevents errors by ensuring `.remove()` is only called when the item is actually found. This ensures only the appropriate IPs are removed, and the script remains error-free even if the remove list contains unexpected or outdated entries.

Update the file with the revised list of IP addresses

After the list is filtered, the updated IPs are converted back into a string, placing each one on a new line:

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

The script then opens the original file in write mode to replace its contents with the cleaned-up list:

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

Using `"w"` mode clears out the old data and replaces it with the updated list, ensuring the allow list reflects the most current access permissions.

Summary

This Python script provides a clean, consistent way to manage IP-based access control by:

- Reading IP addresses from an allow list file
- Converting those IPs into a list for easy handling
- Checking and removing any addresses found in a remove list
- Rewriting the allow list file with only the valid entries

This process cuts down on manual edits, reduces the risk of errors, and keeps the system's access list secure and up to date.