Algorithm for file updates in Python

Project description

The purpose of this project is to control access to restricted content and automate the process, reducing manual and human effort. The algorithm parses a file containing IP addresses allowed to access restricted content, and it removes addresses that should no longer have access. I wrote a Python script to automate the process of parsing the file and keeping it up-to-date.

Open the file that contains the allow list

```
# Open the file that contains the allow list

import_file = "allow_list.txt"

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

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

I begin by opening the “allow_list.txt” file, which contains the IP addresses. The variable import_file is used to store the contents of the file. A separate remove list contains the IP addresses that need to be deleted from the allow list. The text file is opened using the with keyword and the open function with the "r" (read) mode.

Read the file contents

```
: # Read the file contents

  import_file = "allow_list.txt"

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

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

  print(ip_addresses)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
```

To read the contents of the file, the .read() method is used, which stores the data in the variable ip_addresses. The print() function is then used to display the content for inspection.

Convert the string into a list

```
# Convert the string into a list

import_file = "allow_list.txt"

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

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

ip_addresses = ip_addresses.split()

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.170', '192.16
8.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.52.37', '192.
168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

Next, the .split() method is used to convert ip_addresses from a string to a list. By not passing any argument to .split(), it defaults to separating the string based on whitespace. This list format makes it easier to navigate through the data.

Iterate through the remove list

```
# Iterate through the remove list

import_file = "allow_list.txt"

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

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

ip_addresses = ip_addresses.split()

for element in ip_addresses:
    print(element)
```

Using a for loop, I iterate over the ip_addresses list. The loop variable, element, represents each IP address in the list. This step helps identify elements that are present in both the remove_list and ip_addresses.

Remove IP addresses that are on the remove list

```
# Remove IP addresses that are on the remove list

import_file = "allow_list.txt"

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

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

ip_addresses = ip_addresses.split()

for element in ip_addresses:

    if element in remove_list:

        ip_addresses.remove(element)

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.16
8.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

Inside the loop, an if statement is used to check if an IP address in the ip_addresses list is in the remove_list. If found, the .remove() method is called to delete that IP address from ip_addresses.

Update the file with the revised list of IP addresses

```
# Update the file with the revised list of IP addresses

import_file = "allow_list.txt"

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

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

ip_addresses = ip_addresses.split()

for element in ip_addresses:

    if element in remove_list:

        ip_addresses.remove(element)

ip_addresses = " ".join(ip_addresses)

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

  file.write(ip_addresses)
```

After removing the IP addresses, the original file is updated. The ip_addresses list is converted back into a string using the .join() method, which concatenates all elements of the list. I then use another with statement to open the file in "w" (write) mode, replacing its content with the revised IP addresses.

Summary

```
#parses a file

def update_file(import_file, remove_list):

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

    ip_addresses = file.read()

  ip_addresses = ip_addresses.split()

  for element in ip_addresses:

    if element in remove_list:

      ip_addresses.remove(element)

  ip_addresses = " ".join(ip_addresses)

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

    file.write(ip_addresses)

update_file("allow_list.txt", ["192.168.25.60", "192.168.140.81", "192.168.203.198"])

with open("allow_list.txt", "r") as file:

  text = file.read()

print(text)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.3
7 192.168.156.224 192.168.60.153 192.168.69.116
```

This algorithm automates the process of updating an IP address allow list. It demonstrates several useful Python functions:

open(): Opens a file for reading or writing.
.read(): Reads the file's content.
.write(): Writes or appends data to a file.
.split(): Converts a string into a list based on a delimiter (whitespace by default).
for loop: Iterates through a list of IP addresses.
if statement: Checks conditions and removes elements when necessary.
The combination of these functions helps automate the management of access controls for restricted content, improving efficiency and reducing manual effort.