Algorithm for file updates in Python

Project description

As a security analyst for a healthcare company a file needs to be regularly updated to ensure the necessary employees have access to patient records. The file contains the ip addresses of employees who are granted access. There is an allow list for employees with clearance to access the restricted network. There is also a remove list for employees who no longer have access and need to be removed from 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 import_file variable creates and saves a variable to be used later on in the program. The 'with' function along with 'open' describes the file that will be read from.

Read the file contents

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

# Build `with` statement to read in the initial contents of the file

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

This section of code uses the with function to open the allow_list.txt file and store it as a variable using the .read() method called ip_addresses.

Convert the string into a list

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

# Build `with` statement to read in the initial contents of the file

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

This section of code converts the ip_addresses variable into a list using the .split() method

Iterate through the remove list

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information

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

# Build `with` statement to read in the initial contents of the file

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for i in ip_addresses:

    # Display `element` in every iteration

    print(i)
```

This section of code builds the beginnings of a for loop by iterating over the ip_addresses list

Remove IP addresses that are on the remove list

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

# Build `with` statement to read in the initial contents of the file

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Display `ip_addresses`

print(ip_addresses)
```

The aforementioned for loop is then built upon further by introducing an if conditional statement that iterates over all indexes of the ip_addresses list and the remove_list to remove any ip address from the ip_addresses list that matches any address from remove_list.

Update the file with the revised list of IP addresses

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

# Build `with` statement to read in the initial contents of the file

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

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = " ".join(ip_addresses)

# Build `with` statement to rewrite the original file

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

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

The ip_addresses list is then converted into a string and then overwrites the old allow_list.txt file to the updated version with the remove_list ip addresses deleted.

Summary

The Python code iterates over a list of IP addresses associated with database access and checks each address for clearance. For IP addresses that no longer have authorization, the code removes them from the list to ensure secure database access. Using a loop, it systematically scans each IP address, verifying clearance status against an access control variable. Upon identifying unauthorized addresses, it applies a removal mechanism to eliminate them from the list, thereby safeguarding the database from unauthorized access attempts.