Algorithm for file updates in Python

Project description

[Describe the scenario in your own words.]
I am creating an algorithm that regularly updates files that identifies employees and grants them access to restricted content. This is important to manage correctly because patients' records are personal, confidential, and private. The employees in this log file are allowed access based on their IP addresses, and I will be implementing algorithms to update an allow list for IP addresses to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

"Allow_list.txt" :
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
192.168.90.124
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

Open the file that contains the allow list

[Add content here.]

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

```
# Assigning the text file name to a variable for easy access
import_file = “allow_list.txt”

“””
I use the with keyword here to make sure that Python automatically closes the file once I’m finished with it
“””
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, “r”) as file:
```

Read the file contents

[Add content here.]

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

# Assigning the text file name to a variable for easy access
import_file = “allow_list.txt”

“””
I use the with keyword here to make sure that Python automatically closes the file once I’m finished with it
“””
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, “r”) as file:
    ip_addresses = file.read()  # I am reading the file and am assigning the string contents into the ip_addresses  variable
```

Convert the string into a list

[Add content here.]

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

# Assigning the text file name to a variable for easy access
import_file = “allow_list.txt”

“””
I use the with keyword here to make sure that Python automatically closes the file once I’m finished with it
```

```
“””
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, “r”) as file:
        ip_addresses = file.read()          # I am reading the file and am assigning the string contents into the ip_addresses  variable

ip_addresses = ip_addresses.split() # I am turning the string into a list separated by whitespaces and reassigning the value back into ip_addresses
```

Iterate through the remove list

[Add content here.]

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

# Assigning the text file name to a variable for easy access
import_file = “allow_list.txt”

“””
I use the with keyword here to make sure that Python automatically closes the file once I’m finished with it
“””
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, “r”) as file:
        ip_addresses = file.read()          # I am reading the file and am assigning the string contents into the ip_addresses  variable

ip_addresses = ip_addresses.split() # I am turning the string into a list separated by whitespaces and reassigning the value back into ip_addresses

for element in remove_list:
        print(element) # iterating through the remove_list and printing to stdout
```

Remove IP addresses that are on the remove list

[Add content here.]

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

# Assigning the text file name to a variable for easy access
import_file = “allow_list.txt”

“””
```

```
I use the with keyword here to make sure that Python automatically closes the file once I'm finished with it
"""
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, "r") as file:
        ip_addresses = file.read()              # I am reading the file and am assigning the string contents into the ip_addresses  variable

ip_addresses = ip_addresses.split() # I am turning the string into a list separated by whitespaces and reassigning the value back into ip_addresses

for element in ip_addresses: # iterating through ip_addresses
        if element in remove_list: # if the element is in remove_list, then delete it.
                ip_addresses.remove(element) # This is possible because there are no duplicate IP addresses in the ip_addresses list
```

Update the file with the revised list of IP addresses

[Add content here.]

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

# Assigning the text file name to a variable for easy access
import_file = "allow_list.txt"

"""
I use the with keyword here to make sure that Python automatically closes the file once I'm finished with it
"""
# I open the file using the read parameter because I want to see the contents of the file
# I am assigning this opened file into the variable file to be used later
with open(import_file, "r") as file:
        ip_addresses = file.read()              # I am reading the file and am assigning the string contents into the ip_addresses  variable

ip_addresses = ip_addresses.split() # I am turning the string into a list separated by whitespaces and reassigning the value back into ip_addresses

for element in ip_addresses: # iterating through ip_addresses
        if element in remove_list: # if the element is in remove_list, then delete it.
                ip_addresses.remove(element) # This is possible because there are no duplicate IP addresses in the ip_addresses list

# I am converting the list back into a string separated by newline chars and reassigning it into
```

```
# the ip_addresses variable
lp_addresses = “\n”.join(ip_addresses)

# Then I will reopen the import_file txt file and write the contents of ip_addresses into it,
overwriting the previous contents.
with open(import_file, “w”) as file:
        file.write(ip_addresses)
```

Summary

[Add content here.]

In conclusion, I created an algorithm that takes an text file that contains log information that periodically updates the users in the allowed list to make sure that there is zero trust and only allows the lowest level of access in a hospital scenario. This is important because patient information is confidential and this is enforced by law. Therefore, a security company could face charges because they haven’t been following specific laws and guidelines set by the government.

In this algorithm, I opened specific files, read them, converted the string into a list, altered specific elements in the list and removed them, converted the list back into a string, and updated the original file with the new contents of the string. This is powerful because I can control who has access to certain things in the hospital.