Algorithm for file updates in Python

Project description

I´m a security professional working at a health care company. As part of my job, ´M required to regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

I need to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, I should remove those IP addresses from the file containing the allow list.

Open the file that contains the allow list

For this part of the work, I must first explain Python syntax. Python uses a very user-friendly syntax for those who have worked with other programming languages (Java, C++, C#, etc.), so the commands, functions, and keywords may be easier to use in these analysis cases. Having said that, we first need to know the name of the file we are importing and the name of the list containing all the unauthorized IPs. For this first task, we import the file named "allow_list.txt" and define the list containing the IPs we need to remove. Once we have this, we need to convert the contents of that file into a format readable by Python. Therefore, we use `with open (import file, "r") as file:`

The with statement is used with the .open() function in read mode to open the allow list file for the purpose of reading it, In this case, "r" indicates that I want to read it.

```
# 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"]

# First line of `with` statement

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

Read the file contents

Now that we have the code to pass the information to a String, we need to read the file's contents. For this, we used the. read() method to store the information in a variable called 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()

# Display `ip_addresses`

print(ip_addresses)
```

Convert the string into a list

Perfect, now for easier manipulation of the information, as well as to make the data stored in the ip_addresses variable more readable, we will use the .split() method to give me the data in an ordered list. In this algorithm, the .split() function takes the data stored in the variable ip_addresses, which is a string of IP addresses that are each separated by a whitespace, and it converts this string into a list of IP addresses.

By default, the `.split()` function splits the text by whitespace into list elements like you can see in the photo.

```
# 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)
```

Iterate through the remove list

Okay, now we need to navigate through the list to find matching IPs and remove them from the original list. To do this, we need to use an Iterate function to iterate through the information. Specifically, we need to incorporate a for loop, using `ip_addresses` as a parameter and, in this case, calling the `element` variable.

```
# 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:

    # Display `element` in every iteration

    print(element)
```

The keyword `in` indicates to iterate through the sequence `ip_addresses` and assign each value to the loop variable `element`.

Remove IP addresses that are on the remove list

To remove IP addresses from the remove list, we must use a conditional statement to verify that the data currently held by the variable "element" matches an IP address in the remove list. For this, we use an if statement that demonstrates that if "element" is in the remove list, then "element" is removed from the ip_addresses list. In that case, we use the .remove(element) method.

```
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)
```

First, within my `for` loop, I created a conditional that evaluated whether or not the loop variable `element` was found in the `ip_addresses` list. I did this because applying. remove `()` to elements that were not found in ip_addresses would result in an error.

Then, within that conditional, I applied `.remove()` to `ip_addresses`. I passed in the loop variable `element` as the argument so that each IP address that was in the `remove_list` would be removed from `ip_addresses`.

Update the file with the revised list of IP addresses

The work is almost finished; however, the original file still needs to be updated with the allowed IPs. To do this, we must update the original file that was used to create the ip_addresses list. A line of code containing the .join() method has been added to the code so that the file can be updated. This is necessary because ip_addresses must be in string format when used inside the with statement to rewrite the file. Basically, what .join() does (in this case) is convert the information back into a string in order to modify it.

```
# 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)
```

When it becomes a String again, we repeat the process to open the file, with the change that we will not read "r", but we will write or overwrite "w" with the information stored in ip_addresses. I can call the `.write()` function in the body of the `with` statement. The `.write()` function writes string data to a specified file and replaces any existing file content.

Summary

I was able to create an algorithm that removes IP addresses identified in a variable called `remove_list` from the file `allow_list.txt`. This algorithm involves opening a folder, converting it to a string for reading, and then converting that string into a list and storing it in the variable `ip_addresses`. Then, using a loop within `remove:list`, I evaluated each element that was part of `ip_addresses` (the list). If the condition was met, the `.remove()` method was used to remove that element. Finally, `.join` was used to convert `ip_addresses` back into a string to write over the contents of `allow_list.txt`.