Algorithm for file updates in Python

Project description

In this scenario, I'm tasked with creating a python algorithm that identifies allowed users based on IP addresses in the allowed list. Additionally, users with an IP address in the remove list must 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"
```

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

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

The file **"allow_list.txt"** is assigned to the variable **import_file**. A **with** statement is used to open and store the "allow_list.txt" file inside the **file** variable.

Read the file contents

```
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()
```

Here, **.read( )** converts the contents of our file as a string, which is stored inside the **ip_addresses** variable.

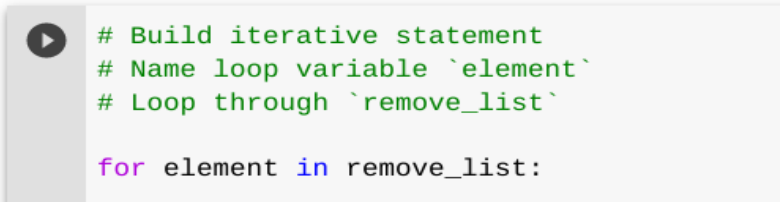
Convert the string into a list

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

ip_addresses = ip_addresses.split()
```

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. To do this, we use the **.split( )** method to convert the **ip_addresses** string into a list.

Iterate through the remove list



Next the header of a **for** loop is set up that iterates through the **remove_list**. **Element** is used as the loop variable.

Remove IP addresses that are on the remove list

```
for element in remove_list:

  # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

      # use the `.remove()` method to remove
      # elements from `ip_addresses`

        ip_addresses.remove(element)
```

The body of our iterative statement contains code that will remove all the IP addresses from the allow list that are also on the remove list.

- First, a conditional statement is created to evaluate if the loop variable **element** is part of the **ip_addresses** list.
- Then, within that conditional, we apply the **.remove(element)** method to the **ip_addresses** list, which removes the IP addresses identified in the loop variable **element**.

Update the file with the revised list of IP addresses

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

ip_addresses = "\n".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)
```

Now that we removed these IP addresses from the **ip_addresses** variable, the algorithm needs to be completed by updating the "allow_list.txt" file with this revised list. First, the **ip_addresses** list needs to be converted back into a string using the **.join( )** method before we overwrite the file. Apply **.join( )** to the string **"\n"** in order to concatenate the elements in the file and put them on a new line.
Then, use another **with** statement and the **.write( )** method to write over the file assigned to the **import_file** variable.

Summary

The above steps allowed me to successfully create a python algorithm that removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of approved IP addresses. This algorithm involved:

1. Opening the file, converting it to a string to be read, and then converting this string to a list stored in the variable `ip_addresses`.

2. Then, python was instructed to iterate through the IP addresses in `remove_list`. With each iteration, I evaluated if the element was part of the `ip_addresses` list. If it was, the `.remove( )` method was applied to the list to remove the element from `ip_addresses`.

3. Afterwards, I used the `.join( )` method to convert the `ip_addresses` back into a string so that I could write over the contents of the `"allow_list.txt"` file with the revised list of IP addresses.