Algorithm for file updates in Python

Project description

In the organization for this project, the access is restricted to an allow IP address, together with a remove list that needs to be updated. The file for allow IP address is identified as `"allow_list.txt".` I created an algorithm to automate updating the `"allow_list.txt"` file and remove these IP addresses that should no longer have access.

Open the file that contains the allow list

First, I assigned the file `allow_list.txt` into string data in variable `import_file.`

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

import_file = "allow_list.txt"
```

I also create another variable for the remove list

```
# 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"]
```

Then I used the `with` statement to open the file

```
# Built "with" statement to open the file

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

The `with` statement is used together with the `open()` function in a read mode to open the list file with a purpose of reading it. In the code `with open(import_file, "r") as file:`, the `open()` function has two parameters. The first identifies the file to import, and then the second indicates what I want to do with the file. In this case, `"r"` indicates that I want to read it. The code also uses the `as` keyword to assign a variable named `file`; `file` stores the output of the `.open()` function.

Read the file contents

I used the `read()` method to read to convert into a string

```
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()
```

The `.read()` method converts the file into a string and allows me to read it. I applied the `.read()` method to the `file` variable in the `with` statement. Then, I assigned the string output of this method to the variable `ip_addresses`.

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 work with the data, it is changed into a list data using the `.split()` method. The purpose of this method is to make it easier to remove the unwanted IP address later on. 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. To store this list, I reassigned it back to the variable `ip_addresses`.

Iterate through the remove list

Next is to iterate through the IP addresses that are elements of the remove list. I used a `for` loop for this.

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

for element in ip_addresses:
```

The `for` keyword starts the `for` loop. It is followed by the loop variable `element` and the keyword `in`. 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

My algorithm requires removing any IP address from the allow list, `ip_addresses`, that is also contained in `remove_list`.

```
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)
```

I created a conditional that evaluated whether or not the loop variable `element` was found in the `ip_addresses` list. 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

Lastly, I need to update the allow list file with the updated list of Ip addresses. To do this, the list must be converted back into a string data

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

ip_addresses = " ".join(ip_addresses)
```

The method `.join()` is used in the code, to combine the list into a string. It is used with the `" "`. The method is applied to the string `" "`, which contains just a space character. The argument of the `.join()` method is `ip_addresses`

Then, I used another `with` statement and the `.write()` method to update the file:

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

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

  file.write(ip_addresses)
```

I used a second argument of `"w"` with the `open()` function in my `with` statement. With this I want to open a file to write over its contents. When using this argument `"w"`, 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. I passed in the `ip_addresses` variable as the argument to specify that the contents of the file specified in the `with` statement should be replaced with the data in this variable.

Summary

I created an algorithm that removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of approved IP addresses. It involves the opening and reading of file, converting it into a list to be iterated into a for loop function. Then conditionals are added into the algorithm to allow only the desired elements to be in the allowed list. The `.remove()` method was used to remove the unwanted ip addresses. After this, 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.