Update a file through a Python algorithm

Project description

At my organization, access to restricted content is managed through an allow list of approved IP addresses stored in the allow_list.txt file. A separate remove list identifies IP addresses that should no longer have access. I developed an algorithm to automate updates to the allow list by removing any IP addresses that appear in the remove list.

Open the file that contains the allow list

In the first part of the algorithm, I opened the allow_list.txt file. I began by storing the file name as a string in the variable import_file.

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

import_file = "allow_list.txt"
```

Then, I used a with statement to open the file:

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

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

In my code, I used a with statement along with the open() function in read mode to access the allow list file. Opening the file allows me to read the IP addresses stored inside it, and the with statement ensures the file is automatically closed once the block is exited. In the code with open(import_file, "r") as file:, the open() function takes two parameters: the first specifies the file to open, and the second—"r"—indicates that the file should be read. The as keyword assigns the opened file to the variable file, which I then use to work with the file's contents inside the with block.

Read the file contents

To read the contents of the file, I used the .read() method, which converts the data 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()
```

I can call the.read() method in the with statement's body when I use an.open() function that takes the argument "r" for "read." I can read the file by using the.read() method, which transforms it into a string. I used the.read() function on the file variable mentioned in the with expression. I then set the variable ip_addresses to the string output of this procedure.

This code scans the contents of the "allow_list.txt" file into a string format, which I can then utilize in my Python program to arrange and extract data.

Convert the string into a list

To remove individual IP addresses from the allow list, I needed the data in list form. So next, I used the .split() method to convert the ip_addresses string into a list.

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

ip_addresses = ip_addresses.split()
```

Appending it to a string variable calls the.split() method. It operates by creating a list from the contents of a string. To facilitate the removal of IP addresses from the allow list, ip_addresses is divided into a list. By default, the.split() function divides the text into list elements based on whitespace. The.split() function in this algorithm creates a list of IP addresses from the data contained in the variable ip_addresses, which is a string of IP addresses separated by whitespace. I returned this list to the variable ip_addresses in order to store it.

Iterate through the remove list

Iterating through the IP addresses that are components of the remove_list is a crucial component of my approach. I used a for loop to accomplish this:

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

for element in remove_list:
```

Python's for loop repeats code for a specified sequence. Applying particular code instructions to every element in a series is the general goal of the for loop in a Python program such as this one. The for loop is initiated by the for keyword. The loop variable element and the keyword in come after it. The loop variable element will be assigned each value when it iterates over the sequence ip_addresses, as indicated by the keyword in.

Remove IP addresses that are on the remove list

Any IP address that is included in both remove_list and the allow list, ip_addresses, must be removed according to my method. I was able to accomplish this using the following code since ip_addresses did not contain any duplicates:

```
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)
```

I started by creating a conditional inside my for loop that determined whether the loop variable element was present in the ip_addresses array. I took this action because using.remove() on components that weren't present in ip_addresses would generate an error.

I then used.remove() on ip_addresses inside that conditional. In order to remove every IP address in the remove_list from ip_addresses, I entered the loop variable element as the input.

Update the file with the revised list of IP addresses

I had to add the updated list of IP addresses to the allow list file as the last step in my method. I have to first turn the list back into a string in order to accomplish this. For this purpose I used the.join() method:

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

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

The .join() method combines all items in an iterable into a single string. It is called on a string that defines how the elements will be separated once joined. In this algorithm, I used .join() to convert the ip_addresses list back into a string so it could be passed as an argument to the .write() method when updating the allow_list.txt file. I used the string '\n' as the separator to ensure that each IP address appears on its own line.

Next, I used another with statement along with the .write() method to update the file:

```
  # 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)
```

In this step, I used the 'w' argument with the open() function inside the with statement. This argument opens the file in write mode, allowing me to overwrite its contents. When a file is opened with 'w', I can call the .write() method within the with block. The .write() method writes the provided string data to the file and replaces any existing content.

In this step, I wrote the updated allow list back to the allow_list.txt file as a string. This ensures that any IP addresses removed from the list no longer have access to the restricted content. To update the file, I called the .write() method on the file object defined in the with statement and passed the ip_addresses variable as the argument. This replaced the file's existing contents with the updated data.

Summary

I constructed an algorithm that eliminates IP addresses from the "allow_list.txt" file of authorized IP addresses that are listed in a remove_list variable. Using this approach, the file was opened, transformed into a readable string, and then transformed into a list that was kept in the variable ip_addresses. The IP addresses in remove_list were then iterated through by me. I examined if the entry was included in the ip_addresses list for every iteration. If so, I removed the element from ip_addresses using the.remove() method. In order to copy the updated list of IP addresses across the contents of the "allow_list.txt" file, I then used the.join() technique to transform the ip_addresses back into a string.