Algorithm for file updates in Python

Project description

At my organization, access to restricted content is controlled with an allow list of IP addresses. The "allow_list.txt" file identifies these IP addresses. A separate remove list identifies IP addresses that should no longer have access to this content. 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

For the first part of the algorithm, I opened the "allow_list.txt" file. First, I assigned this file name as a string to the import_file variable:

```
# 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:
```

Read the file contents

In order to read the file contents, I used the .read() method to convert it into the string.

```
# Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

ip_addresses = file.read()
```

Convert the string into a list

In order to remove individual IP addresses from the allow list, I needed it to be in list format. Therefore, I next used the .split() method to convert the ip_addresses string into a list:

```
# 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()
```

Iterate through the remove list

A key part of my algorithm involves iterating through the IP addresses that are elements in the `remove_list`. To do this, I incorporated a `for loop`:

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

for element in ip_addresses:
```

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`. Because there were not any duplicates in `ip_addresses`, I was able to use the following code to do this:

```
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)
```

Update the file with the revised list of IP addresses

As a final step in my algorithm, I needed to update the allow list file with the revised list of IP addresses. To do so, I first needed to convert the list back into a string. I used the .join() method for this:

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

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

Then, I used another with statement and 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)
```

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. This algorithm involved opening the le, converting it to a string to be read, and then converting this string to a list stored in the variable ip_addresses. I then iterated 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, I applied the .remove() method to it to remove the element from 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.