Algorithm for file updates in Python

Project description

The organization I am working for needs to audit access control privileges. This will be accomplished by checking the allow list file against another list of people to remove to make sure it's updated. I made an algorithm to update the allowed list and to remove the access of people no longer on the list.

Open the file that contains the allow list

At the beginning of the algorithm, I store the filename in a **variable** and then open it using the **with** statement and **open** function. This is done with the argument set to read and lets me access the contents of "allow_list.txt".

```
# 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:
```

Read the file contents

To view the contents of the file I use the **read** command while inside the body of the **with** statement, on the variable we created earlier to open it. During this process it is stored in a variable called **ip_addresses**. By storing the data we want in ip_addresses I can organize and use it later on.

```
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()
```

Convert the string into a list

This part of the project changes the data type of the allow_list.txt file contents from one large string to a list where each element is a different ip address. This is done using the **split** function which assumes the delimited character is a space if not specified. The result is a list of string elements separated correctly due to the original format of the file. Also it does not create a new variable to store this list but change the original variable to become one.

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list

To check the remove list against the allow ip addresses, I iterated through it using a **for** loop. The loop repeats the body code until I have iterated over the entire list which is what the in keyword **in** is for. This means I can compare each element of the remove list to every allow list element.

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

for element in remove_list:
```

Remove IP addresses that are on the remove list

Inside the for loop is code that is executed with every new iteration. Meaning an **if statement** is checked for every element of the remove list. If an element of the remove list also exists in the allow list, then it must be removed from the allow list. This bottom level code is only executed on those conditions, updating the list to the most recent access control. The ip address is removed using the **remove** function and specifying the specific element.

```
for element in remove_list:

    if element in ip_addresses:

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

        ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses

The last part of the algorithm is to take the updated allow list and convert it back to the format it was in the text file. Then to put that newly formatted variable back into the allow_list.txt. The **join** function is used for this to change it from a list back to one large string. The join is made using a new line symbol so each element exists on a different line of the text file.

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

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

Now to write this variable to allow_list.txt I can modify the code I used to open and read the file to write to it instead. This is done by changing the **open** function argument from “**r**” to “**w**”. At that point I used the **write** function of the file **object** while in the with body to replace the content of the file. At this point the allowed ip addresses are correct and updated.

```
# 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

This algorithm takes in the unmodified allowed list of ip addresses and compares it to an updated list of removed addresses. Any addresses that exist on both lists are removed from the allowed list of addresses. This results in the most updated list of allowed addresses. The tools used are the python file input, and read or write functionality as well as its split and join functions to manipulate the string and list variables used.