Algorithm for file updates in Python

Project description

For this project, I will be creating an algorithm using Python script to remove IP addresses from a list of authorized IP addresses in an organization. Authorized IP addresses are contained in a file named `allow_list.txt`.

Open the file that contains the allow list

Authorized IP addresses in the organization are contained in the `allow_list.txt`. First, I assigned the IP addresses to be removed under the variable, `remove_list`, for ease of access.

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

import_file = "allow_list.txt"

# 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"]
```

Afterwards, I commanded Python to open the `allow_list.txt` for the purpose of reading it as a file:

```
# First line of `with` statement

with open(allow_list.txt, "r") as file:
```

Read the file contents

To initialize the read, I use the `.read()` method on the file and assign it to a new variable known as **"ip_addresses"**.

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

import_file = "allow_list.txt"

# 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"]

# 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()
```

The with keyword commands Python to interact with the subsequent command once. The open() function will open the import_file variable and "r" is used as the second argument to indicate the instruction to read the variable as a file. In the next line, the file will be read using the .read() method then converts its contents into a new variable known as ip_addresses.

Convert the string into a list

Next, I convert the IP addresses under the allow_list.txt from strings into lists using the .split() method. This will turn each IP address into their own individual element for ease of use.

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

import_file = "allow_list.txt"

# 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"]

# 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

Now, I command Python to inspect each element on the newly created list of IP addresses using an iterative for loop statement.

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

import_file = "allow_list.txt"
```

```
# 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"]

# 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()

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

for element in ip_addresses:
```

Now, I command Python to inspect each element on the newly created list of IP addresses using an iterative for loop statement. The for keyword will run a loop using a loop variable that was created called element. The in keyword will indicate that the loop will occur in the ip_addresses variable

Remove IP addresses that are on the remove list

Now that each element is being iterated, I can now run the conditional that would remove the IP address from the allow_list.txt if it can also be found in the remove_list.

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

import_file = "allow_list.txt"

# 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"]

# 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()

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

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)
```

If the same IP address can be found in both the remove_list and ip_addresses then it will be removed from ip_addresses using the .remove() method. Within the .remove() method, the element is passed as a parameter to indicate what needs to be removed and element, as we know, are the IP addresses in the list now.

Update the file with the revised list of IP addresses

WIth the IP addresses removed, I will now write the new ip_addresses into the allow_list.txt.

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

import_file = "allow_list.txt"

# 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"]

# 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()

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

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)

# 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)
```

The “/n” next to the .join() method will convert each element, each IP address, into a new line. The .join() method will be used to convert the contents of ip_addresses from the iterable, the earlier “for element in ip_addresses”, from a list back into a string. Afterwards, I used another with keyword followed by an open() function. The first argument is import_file which indicates the variable to interact with “w” as the second argument to instruct Python to write the file.

Summary

I have created an algorithm that will remove IP addresses off a master list if they are no longer authorized to have access to the organization’s data.

- First, I assign the master list, allow_list.txt, to the variable called import_file.
- Then, I read the import_file using a with keyword then store the read file into a new variable called ip_addresses.
- Afterwards, I converted the string contained in ip_addresses into a list that separates each IP address into its own individual element.
- I create an iterative to command Python to identify each IP address as element

- Next, I created a condition within the interactive that will remove an element if it can be found in both `ip_addresses` and `remove_list`
- Finally, I will convert the new `ip_addresses` back into a string using the `.join()` command with `“/n”` to separate each element into a new line. Now that `ip_addresses` is a string, it can be written back into the `import_file` using the with keyword and `open()` function along with the `“w”` to pass a write argument.