Algorithm for file updates in Python

Project description

Using Python to create an algorithm that checks whether or not an IP address is allowed. If it is not, it will be removed from a list of allowed IP addresses.

Open the file that contains the allow list

```
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
```

```
import_file = "allow_list.txt"
```

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

Here I am using 'with open()" to specify the file I want to open, in this case "allow_list.txt" which is stored in the import_file variable.
"r" tells this python to read the file. This will be stored in the variable "file".

Read the file contents

```
ip_addresses = file.read()
```

To make the file easier to read, I am using the 'read()' function, converting the text into a string, and storing this into a variable named ip_addresses for easy access.
Then, telling Python to display the output.

```
print(ip_addresses)
```

Convert the string into a list

Next I convert the previous string into a list, again making it easier to read and work with further.

```
ip_addresses = ip_addresses.split()
```

To do this, I am using the '.split()' function. This will take the information and convert each element into a list, rather than one group of text. By default, it is separating the data where there is any white space.

Iterate through the remove list

Here, I am setting up a “for” loop that will iterate through each element to determine which IP address does or does not belong.

```
for element in ip_addresses:
```

Remove IP addresses that are on the remove list

Now, using an ‘if’ statement, I am removing any element from the ip_address variable that is found in the remove_list.
‘Element” is a variable created to signify an element of the list, in this case an IP address.
‘Element’ is being passed into the ‘remove’ function.

```
if element in remove_list:
ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses

Using the “join()” function, I can update the existing list of IP addresses into a string.

```
ip_addresses = " ".join(ip_addresses)
with open('ip_addresses' , "w") as file:
```

Next I am updating the list with the new information being returned using “with open()” again, this time with “w” as the second parameter. “w” in this case stands for write, so I am adding to the existing file rather than just reading.

```
file.write(ip_addresses)
```

Lastly, using “file.write()” to update ‘ip_addresses’.

Summary

To effectively write this algorithm, a few core functions were needed. ‘Open’ is used to tell python what files to open and what to do with them. Though, this doesn’t display anything, so I must follow up with the “print()” function if I want to verify that it is returning the correct information. Typically, I always want to see; so I used “print()” quite often. In a real working scenario, I would define this function so that I can call back to it whenever needed.