Algorithm for file updates in Python

Project description

This project is centered around the fundamentals needed to sort out an “approved_list” of ip addresses in Python utilizing an algorithm that iterates through a text file of ip addresses and compares that list of ip addresses to a “remove_list” of ip addresses. The algorithm then removes any corresponding ip addresses from the text file and sends the rest on over to stay on the “approved_list”

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

```
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.
↪58.57"]
```

Open the file that contains the allow list

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

The allow_list.txt file is stored in the import_file variable. Then comes the with statement which will close the function once it's done and no further indentions follow. Next follows the open function which in Python allows you to open a file and to read or write over the existing file. The first parameter is import_file which is the variable containing the file the second is “r” which represents “read” and indicates how the file should be handled. After indicating the file and how it is going to be handled I now store it in a container called “file”

Read the file contents

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

Now I create a new variable called “ip_addresses” to store what the “file.read()” using the .read() method

Convert the string into a list

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

Next, I update the file contents by using the “.split()” method which separates the contents of a string and turns them into a list with a specified character. Since I did not pass through any

argument into the parenthesis the character that divides each element in the list defaults to a "," comma.

Iterate through the remove list

```
for element in ip_addresses:
    if element in remove_list:
```

Now that each element in the list is separated by a "," comma. I can use a for loop to iterate through the ip_addresses variable. I name each item in the loop "element". Next, I start an "if" conditional statement which tells Python that for each iteration if the current element in the ip_addresses list is also in the "remove_list" to complete an action.

Remove IP addresses that are on the remove list

```
ip_addresses.remove(element)
```

The action we tell Python to complete for each corresponding element is to use the .remove() method. Removing it from the list.

Update the file with the revised list of IP addresses

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

Next, in order to rewrite a file it has to be in string format so I use the .join() method to turn the argument into a string. First I specify how I want the elements in the list separated once placed into a string. I want them separated by a space so I place it in quotation marks to keep it in string format. After that I insert the argument "ip_addresses" which is the variable that contains the file. The argument is placed inside the parenthesis.

Summary

Placing an algorithm like this inside a function or incorporating this algorithm into a function can save massive amounts of time and code. All the while creating time for more important tasks rather than sorting through and comparing 2 lists and removing selected elements and/or adding selected elements. Then updating the file containing the revised list. Learning how to speed up and simplify tasks will strengthen the efficiency of a team and overall productiveness.