Algorithm for file updates in Python

Project description

In this project, the control of who can access restricted content is controlled with an allow list of IP addresses. There is also a remove list of IP addresses that should no longer have access to this restricted content. I created an algorithm to erase the IP addresses in the allow list that are in that remove list.

Open the file that contains the allow list

To start, I assigned, as a string, the file “allow_list.txt” into the import_file variable:

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

Then, I opened the file with the with statement:

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

In order to properly open a file in Python, I needed to use the keyword with, the function open() with its two parameters, and the keyword as followed by a variable name.
The with keyword handles errors and manages external resources when used with other functions. In this case it’s used with the open() function. It will then manage the resources by closing the file after exiting the with statement.
The open() function opens a file in Python. As mentioned before, open() function as two parameters separated by commas, the first is the file we want to open and the second is what we want to do with the file. The second parameter can be either “r”, which indicates that we want to read the file, “w” if we want to write to the file and “a” if we want to append to a file.
When we open a file using with open(), we must provide a variable that can store the file while we are within the with statement. We can do this through the keyword as followed by a variable name, in this case the variable is file. file stores the output of the open() function.

Read the file contents

After the use of with open(import_file, “r”) as file: to import “allow_list.txt” into the file variable, we have to indicate what to do with the file on the indented lines that follows it:

```
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 .read() method converts files into strings. This is necessary in order to use and display the contents of the file that was read. In this example, the file variable is used to generate a string of the file contents through .read(). The string is then stored in another variable called ip_addresses.
In summary, this code will read the content in the “allow_list.txt” file into a string format.

Convert the string into a list

In order to remove IP addresses from the allow list, I needed it to be in a list format. To achieve this, I used the .split() method:

```
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()
```

The split() method converts a string into list. It separates the string based on a specified character that’s passes into .split() as an argument. When appended to a variable, .split() converts the string into a list, making it easier to remove elements of it.

Iterate through the remove list

An important part of this project is to iterate through the list of IP addresses stored in a remove_list variable:

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

To achieve this, I used a for loop:

```
for element in remove_list:

    print(element)
```

```
192.168.97.225
192.168.158.170
192.168.201.40
192.168.58.57
```

The for loop repeats code for a specified sequence. The for loop consists of the for keyword, a loop variable, and the sequence the loop will iterate through. The keyword for tells Python to start a loop. The loop variable can have any name that we want, as it is a temporary variable that is used to control iterations of a loop, in this example is element. The keyword in tells Python to iterate through the remove_list variable and assign each value to the loop variable element.

Remove IP addresses that are on the remove list

My algorithm requires removing the ip addresses in the allow list that are stored in the remove_list variable. Since there are no duplicates in "allow_list.txt", I was able to use the following code:

```
for element in remove_list:

  # Build conditional statement

    if element in ip_addresses:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)
```

In the body of this iterative statement, I created a conditional that evaluates if the content of the element variable is found in the ip_addresses variable.

Then, within that conditional, we applied the .remove() method to ip_addresses. I passed the element variable as an argument, so that each IP address that was in remove_list would be removed from ip_addresses.

Update the file with the revised list of IP addresses

As a final step. I needed to update the allow list with the revised list of IP addresses. In order to update the "allow_list.txt", first I converted the ip_addresses variable into a string with the .join() method:

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

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

The .join() method concatenates the elements of an iterable into a string. The syntax used with .join() is distinct from the syntax used with .split() or .remove(). In methods like .split() or .remove(), we append the method to the string or list that we are working with and then pass in other arguments. However, with .join(), we must pass the list that we want to concatenate into a string as an argument.
The "\n" character indicates to separate the elements by placing them on new lines.

I used the .join() method to be able to pass ip_addresses as an argument to the .write() method when writing to the file "allow_list.txt" with the open() function in my with statement:

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

  # Rewrite the file, replacing its contents with `ip_addresses`

    file.write(ip_addresses)
```

This time, the with statement has "w" as a second argument, which indicates that we want to write to the file. With the "w" argument, I can call the .write() function in the body of the with statement. The .write() function writes string data to a specified file and replaces any existing content. As a result, the content of ip_addresses variable was written to the "allow_list.txt".

Summary

I created an algorithm that erases the IP addresses identified in a remove_list variable from the "allow_list.txt" file. This algorithm involved opening a file, converting it to a string to be read, then converting this string to a list in the variable ip_addresses. Then, I iterated through the IP addresses in remove_list. With each iteration, I evaluated if the element was part of the ip_addresses. If it was, I applied the .remove() method to remove the element from ip_addresses. After this, I used the .join() method to convert the ip_addresses back into string so that I could write over the contents of the "allow_list.txt" file with the revised list of IP addresses.