Update a file through a Python algorithm

Project description

At organization XYZ access to restricted content is controlled with an allow list of IP addresses. The `"allow_list.txt"` file identifies these IP addresses. A separate remove list identifies IP addresses that need to be removed from the allowed list. An algorithm was developed to update the `"allow_list.txt"` file by removing these IP addresses that should no longer have access.

Open the file that contains the allow list

Assign the `"allow_list.txt"` file as a string to the `import_file` variable:

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

import_file = "allow_list.txt"
```

A `with` statement to open the file:

```
# Build `with` statement to read in the initial contents of the file

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

The `with` statement is used with the `.open()` function in read mode to open the allow list file for the purpose of reading it. The purpose of opening the file is to allow me to access the IP addresses stored in the allow list file. The `with` keyword will help manage the resources by closing the file after exiting the `with` statement.

In the code `with open(import_file, "r") as file:`, the `open()` function has two parameters. The first identifies the file to import, and then the second indicates what I want to do with the file. In this case, `"r"` indicates that I want to read it. The code also uses the `as` keyword to assign a variable named `file`; `file` stores the output of the `.open()` function.

Read the file contents

The `.read()` method was used to convert the file into string format.

```
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 string output of the read() method is stored in the variable `ip_addresses`.

In summary, this code reads the contents of the `"allow_list.txt"` file into a string format that allows me to later use the string to organize and extract data in my Python program.

Convert the string into a list

In order to remove individual IP addresses from the allow list, the `.split()` methodwas used to convert the `ip_addresses` string into a list:

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

ip_addresses = ip_addresses.split()
```

The `.split()` function is called by appending it to a string variable. It works by converting the contents of a string to a list. The purpose of splitting `ip_addresses` into a list is to make it easier to remove IP addresses from the allow list. By default, the `.split()` function splits the text by whitespace into list elements. In this algorithm, the `.split()` function takes the data stored in the variable `ip_addresses`, which is a string of IP addresses that are each separated by a whitespace, and it converts this string into a list of IP addresses. To store this list, it was reassigned back to the variable `ip_addresses`.

Iterate through the remove list

A key part of my algorithm involves iterating through the IP addresses that are elements in the `remove_list`. To do this, I incorporated a `for` loop:

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

for element in remove_list:
```

The `for` loop in conjunction with the loop variable `element` iterates through the remove_list.

Remove IP addresses that are on the remove list

This portion of the algorithm requires removing any IP address from the allow list, `ip_addresses`, that is also contained in `remove_list`.

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

for element in remove_list:

  # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

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

        ip_addresses.remove(element)
```

The conditional `if` statement checks whether or not the loop variable `element` was found in the `ip_addresses` list.

If the statement was TRUE then the value contained in the loop variable `element` was removed using the `.remove()` method from `ip_addresses`.

Update the file with the revised list of IP addresses

As a final step, the allow list file was updated with the revised list of IP addresses. The `.join()` method was used to convert the list back into a string.

```
# 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 combines all items in the variable `ip_addresses` into a string. The string `("\n")` as the separator to instruct Python to place each element on a new line.

Once completed the `.write()` will be used update the file `"allow_list.txt"`.

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

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

  file.write(ip_addresses)
```

For this portion, the second argument of `"w"` with the `open()` function in the `with` statement indicates that a file will be opened to write over its contents. When using this argument `"w"`, the body contains the `.write()` function to write the string data of `ip_addresses` to the specified file `"allow_list.txt"` and replaces any existing file content.

The file is now updated, so that the restricted content will no longer be accessible to any IP addresses that were removed from the allow list.

Summary

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