Algorithm for file updates in Python

Project description

As a security analyst, I'm responsible for developing an algorithm that parses a file containing IP addresses `"allow_list.txt"` that are allowed to access restricted. Variable named `remove_list` that contains the list of IP addresses to be removed.IP addresses need to be removed from the text file.

Open the file that contains the allow list

First I assigned the text file `"allow_list.txt"` into a string variable called import_file

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

import_file = "allow_list.txt"
```

Start by opening the text file using the `import_file` variable, the `with` keyword, and the `open()` function with the `"r"` parameter.

```
# First line of `with` statement

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

Read the file contents

use the `.read()` method to read the imported file and store it in a variable named `ip_addresses` as string

```
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()
```

Convert the string into a list

reading the file, reassign the `ip_addresses` variable so its data type is updated from a string to a list using the `.split()` method.

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list

Build the iterative statement. Name the loop variable `element`, loop through `ip_addresses`, and display each element

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

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

Remove IP addresses that are on the remove list

if the current element in the `ip_addresses` list is in the `remove_list`, the `remove()` method should be used to remove that element.

```
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)
```

Update the file with the revised list of IP addresses

Update the original file that was used to create the `ip_addresses` list. A line of code containing the `.join()` method has been added to the code so that the file can be updated. This is necessary because `ip_addresses` must be in string format when used inside the `with` statement to rewrite the file.

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

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

The argument of the `.join()` method is the iterable you want to convert, and in this case, that's `ip_addresses`. As a result, it converts `ip_addresses` from a list back into a string with a space between each element and the next.

After this line with the `.join()` method, build the `with` statement that rewrites the original file. Use the `"w"` parameter when calling the `open()` function to delete the contents in the original file and replace it

```
# 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)
```

Summary

Developed an algorithm that parses the text file `"allow_list.txt"` of IP addresses and updates the file by removing those addresses in`"remove_list"` that no longer have access to the restricted content.Both an iterative statement and a conditional statement helped to remove the elements of `remove_list` from the `ip_addresses` list. Then the `.join()` method is applied to a string consisting of the character that will be used to separate every element in the iterable once it's converted into a string. Finally build the `with` statement that rewrites the original file. Use the `"w"` parameter when calling the `open()` function to delete the contents in the original file and replace it.