Algorithm for file updates in Python

Project description

In this project, the scenario was that I was a security professional at a healthcare company. Part of my job is regularly updating a file specifying the employees authorized to access restricted content. This is based on who is working with personal patient records. The employees' access is limited by their IP address. There is an allow list of IP addresses that are permitted to sign into the restricted subnetwork, and there is a remove list identifying which employees need to be removed from said allow list. As such, my task was to develop an algorithm in Python that would check whether the allow list contained any of the IP addresses present on the remove list and remove them if so.

Open the file that contains the allow list

To start, I needed to assign appropriate variables with the respective values: the file name for the allow list and the list of IPs to remove. This was done by assignment using the string `"allow_list.txt"` and a list of strings that contained four IP addresses to remove. Then, to open the file, we started the 'with' statement needed, which involves the keyword `with`, the function `open()` and its two parameters, followed by `as` and then the variable to hold the file object, in this case called `file`. The `with` keyword and statement allows us to open a file and interact with it. Once finished, Python will automatically close the file and prevent it from taking up additional resources. The `open()` function requires the file name to open as the first argument and the mode to open the file in, this being either read, `"r"`, or write, where write can be entirely overwriting the current file or opening a new file with `"w"`, or appending to an existing file with `"a"`. The keyword `as` simply tells the `with` statement where to store the output of the `open()` function and what that variable is called, which is simply `file` as mentioned previously.

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

# First line of `with` statement

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

Read the file contents

After opening the file using the `with` statement discussed above, we can now read in the file contents and store them as a string in a variable to allow us to interact with them more easily. This is done by simply calling the `.read()` method on the file and assigning that output to the variable `ip_addresses`, as seen here.

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

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

To better search for an IP address to remove from this allow list, we will convert the string stored in `ip_addresses` to a list of strings using the `.split()` method. This separates the IP addresses in this string into individual list elements based on the whitespaces in the original string.

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list

The next step is to set up a `for` loop to iterate through all the IP addresses in our original allow list, which is now stored as a list in `ip_addresses`. Therefore, we create a `for` loop using `element` as our iterator through `ip_addresses`.

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

for element in ip_addresses:
```

Remove IP addresses that are on the remove list

To remove the specified IP addresses, we will compare each element in our allow list to the elements in the `remove_list`, and if we find a match for the current element, we need to remove that element from `ip_addresses`. Therefore, we call the `.remove()` method on the `ip_addresses` list and pass `element` in as the argument, which tells `.remove()` what item in the list to remove. It should be noted that using `.remove()` in this way is only possible because

there are no duplicates in the `ip_addresses` list. If there were duplicates, it would not work, or we would have to call `.remove()` as many times as there were duplicates. This is because `.remove()` functions by finding the first instance of the item it was told to remove when starting from index 0 in the list and iterating till the end of the list.

```
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

Now that we have removed all IP addresses specified in `remove_list`, we need to update the `allow_list.txt` file to reflect this. In order to write to a file, the data must be in string form, so we first `.join()` the list `ip_addresses` together using the string `"\n"` which results in the IP addresses being joined together into one string, separated by new lines. Then, we must reopen the file in write mode and write over the existing contents with the latest updated contents. We accomplish this by using another `with` statement, `open()` specifying `import_file` and `"w"`, and `as file` again. However, within the `with` statement, we call the `.write()` method on the file and pass our updated string `ip_addresses` as the argument.

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

ip_addresses = "\n".join(ip_addresses)

# 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

In conclusion, I developed a comprehensive algorithm to handle the assigned task. It starts by opening a specified file and obtaining the list of IP addresses to remove. Then, it reads the contents of the opened file and stores these allowed IP addresses to a string variable before splitting the elements into a list. Once in list form, it iterates through all the elements and compares them to the list of IPs to remove, and if a match is found, it removes that element

from the original allowed IPs. After all removals have been completed, it joins the modified IP list together, reopens the file, and overwrites it with the new IP list.