Algorithm for file updates in Python

Project description:

As part of my job as a security professional working at a healthcare company I'm required to update a file that identifies the employees that are greenlighted to access restricted content, based on who is working with personal patient records. This access is restricted based on their IP address, and I have a list of each particular address. I also have a list of employee IP's that I have to remove because they have been deemed unauthorized.
For this I will develop an algorithm with Python to check if the allow list has unauthorized IP's that were previously identified in the remove list.

Opening the file that contains the allow list:

First I imported the previously created list that contained the allowed IP's, "allow_list.txt" and assigned it to the variable import_file. I also did the same for the list of IP's that had to be removed (remove_list) but the algorithm won't need it until later.

```
# 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:
```

After defining the variables for the files, I used the open() function to read it. The first parameter (import_file) is the file that will be read, and the second parameter tells the function to read it ("r"). I also chose to use the as keyword to store the output of .open() in the variable file while it is reading it.

This was defined after a with statement so it manages system resources by closing the file after the algorithm leaves the with statement.

Reading the file contents:

```
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()
```

Within the previously defined `with` statement, I specified the function used to actually read the file. This was done by using the `.read()` method and saving it to a new variable named `ip_addresses`. This new variable will save the contents of the `import_file` into a string that I can further iterate through for subsequent use.

Converting the string into a list:

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

Due to the difficulty of iterating through this particular string, I used the `.split()` method to transform the string into a list. With all of this taken care of, now I can achieve my initial objective of removing the IP's specified in the `remove_list` variable.

Removing IP addresses that are on the remove list:

Using a `for` loop I will iterate through every `element` of the list that has access to the restricted data (`ip_addresses`).

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

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)

# Display `ip_addresses`

print(ip_addresses)
```

Within this loop i added an if statement that checks if the `element` that is being iterated in the `for` loop coincides with an element from the list of unauthorized IP's (`remove_list`), and then if it evaluates to True it afterwards runs the `.remove()` method to exclude that particular `element` from the list of allowed IP's (`ip_addresses`).

After this loop ends, I'm left with a filtered list of authorized `ip_addresses` that are ready to be transformed back into a string so that I can write them into a file.

Updating the file with the revised list of IP addresses:

To convert it into a string I used the `.join()` function. Each entry was separated by a single space (`" "`).

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

ip_addresses = " ".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)
```

Now I have a string ready to be written. To do so I start a `with` statement with an `open()` function in write mode (`"w"`). Within this statement I use the function `.write()` to overwrite the contents of the `ip_addresses` into the original `import_file`.

The newly overwritten `import_file` (`"allow_list.txt"`) is now properly filtered and can be used as a reliable list of allowed IP's.