Algorithm for file updates in Python

Project description

I am a security professional working at a health care company. As part of my job, I am required to regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted based on their IP addresses. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There is also a remove list that identifies which employees I have to remove from the allow list.

Open the file that contains the allow list and read the file contents

```
# 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"]

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

with open(import_file, 'r') as file:
    ip_addresses = file.read()
```

Convert the string into a list

```
# I use the split method to convert the string 'ip_addresses' to a list.
# Since the IP addresses are separated by a space, I don't have to enter an argument.
ip_addresses = ip_addresses.split()
```

Iterate through the remove list and remove IP addresses that are on the remove list

```
# This is a list of IP addresses that need to be removed from the list.
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
# I use a for loop on the ip_addresses list.
# This loops through the entire list for each ip_address,
# which is called 'element' for each loop
for element in ip_addresses:
    # I want to check if the IP address in the current loop is in the remove_list.
    # To do so, I use an if statement that checks whether the element is in the remove_list.
    if element in remove_list:
        # When the if statement is triggered, the element is removed from the ip_addresses list.
        ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses

```
# Now that I've removed the bad IP addresses, I want to put the string back together
# To do so, I use the .join method. This concatenates the entire list into a single string,
# each element separated with the string that I used the .join method on, in this case a space.
ip_addresses = " ".join(ip_addresses)

# I use the with statement again, but this time I use "w" mode for write.
with open(import_file, "w") as file:
    # I then use the .write method to rewrite the file,
    # Replacing its contents with the new string of IP addresses
    file.write(ip_addresses)
```

Summary

I have created an algorithm that removes IP addresses from a file. The code is from a lab I had to do in this course. I can think of a number of improvements:

- Instead of writing the remove_list inside the code, this can be its own file. Then I can add any IP addresses that need to be removed into a file, and then run the code.
- The lab covers how to make the above algorithm into a function. Though the course did not cover that in this activity, this is still a good idea, and I have added a screenshot of the function from the lab below:

```
def update_file(import_file, remove_list):

  # 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()

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

  # 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)

  # 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:
    file.write(ip_addresses)
```