Algorithm for file updates in Python

Project description

For this project, I work as a security professional for a healthcare company. I'm required to regularly update a file that identifies the employees who can access restricted content. There is an allow list for IP addresses permitted to sign into the company's restricted subnetwork. There is also a remove list that identifies which employees I need to remove from the allow list. My task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list.

Open the file that contains the allow list

The file I want to open is called **"allow_list.txt."** I assigned a string containing this file name to the **import_file** variable. Then, I created the **remove_list** variable, which contains a list of IP addresses that are no longer allowed to access restricted information. Then, I used a **with** statement to open the file.

```
# 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

I used the **.read()** method to convert the contents of the allow list file into a string so that I could read them. I stored the string in a variable named **ip_addresses**.

```
# 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

I used the **.split()** method 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()
```

Iterate through the remove list

The **remove_list** variable contains all the IP addresses that should be removed from the **ip_addresses** list. I set up the header of a for loop that will iterate through the **remove_list**. I used **element** as the loop variable to accomplish this.

```
# 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

In the body of the iterative statement, I added code that will remove all the IP addresses from the allow list that are also on the remove list. First, I created a conditional that evaluates if the loop variable **element** is part of the **ip_addresses** list. Then, within that conditional, I applied the **.remove()** method to the **ip_addresses** list, and I removed the IP addresses identified in the loop variable **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

After removing the desired IP addresses from the **ip_addresses** variable, I completed the algorithm by updating the file with the revised list. I converted the **ip_addresses** list back into a string using the **.join()** method. I applied the **.join()** method to the string **"\n"** in order to separate the elements in the file by placing them on a new line. Then, I used another **with** statement and a **.write()** method to write over the file assigned to the **import_file** variable.

```
# 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

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.