Algorithm for file updates in Python

Project description

Working for a health care company as a security professional, I am regularly required to update an allow list file that determines which IP addresses can access patient records on a subnetwork. There is a remove list with IP addresses that cannot access the patient files. I must ensure those IP addresses listed on the remove list are removed from the allow list by creating an algorithm using Python.

Open the file that contains the allow list

First, I assigned the variable to the `allow_list.txt` file.
Then, I used the `with` keyword which handles errors and manages external resources when used with other functions. The `with` keyword also ensures that all resources are released and the file is closed after I read it. This is used with the `open()` function, to open the file. The "r" indicates that we want to read the file. The `as` statement works to reference another object as the variable name. I store the file while using the `with` statement. The code shown here on the last line assigns `file` as the variable name for the output of the `open()` function.

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

import_file = "allow_list.txt"
```

```
# First line of `with` statement

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

Read the file contents

The `.read()` method was used to read the file contents by converting the contents of the allow list file into a string. The `print()` function was used to display the contents, and the variable named `ip_addresses` was passed within the `print()` function.

```
# 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()

# Display `ip_addresses`

print(ip_addresses)
```

Convert the string into a list

The split() method was used to convert the ip_addresses string into a list, in order to remove individual IP addresses from the allow list.

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

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

Iterate through the remove list

The header of a for loop was created that will iterate through the remove_list, using element as the loop variable.

```
# 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

Code was added to the body of the iterative statement that will remove all the IP addresses from the allow_list.txt that are also on the remove_list. Applying the .remove() method is possible here because there are no duplicates in the ip_addresses 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)

# Display `ip_addresses`

print(ip_addresses)
```

Update the file with the revised list of IP addresses

The .join method is used to combine all items in an iterable into a string. I used the “\n” string to have Python separate and place each element on a new line. The “w” is used here to indicate that I want to write to the “import_file”. The with and .write method update the file.

```
# 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

We displayed contents of the allow_list.txt. The "r" read statement and the .read() method were used to read the import_file. The split() method converted the ip_addresses string to a list. An iterative statement was used to loop through a file to find IP addresses matching a condition. The .remove() method was used to remove IP addresses found in that list that were no longer allowed to access the subnetwork. The .write() method was used to write the original file to contain the updated ip_addresses list.