Algorithm for file updates in Python

Project description

I am a security analyst at a healthcare company. I was tasked with creating an algorithm using Python code that parses a series of IP addresses that can access restricted information and removes the addresses that are no longer allowed for a file, “allow_list.txt”, which contains personal patient records.

Open the file that contains the allow list

I first accessed the file that contains the allow list using the following code:

```
# 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:
```

To open the file in question, “allow_list.txt,” I used the **import_file** variable, the **with** keyword, and the **open()** function with the “**r**” parameter.

Read the file contents

I expanded on the previous code using the **.read()** function to store the **ip_addresses** variable:

```
# 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:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read(text)

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.20
3.198 192.168.218.219 192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

The .read() function allowed me to read the contents of the file. The output displayed the IP addresses on the allow list. I found that the four IP addresses on the remove list were on the allow list.

Convert the string into a list

I then converted the allow list into a string using the **.split()** function:

```
# 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:

  # 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()

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

Converting the string into a list was for preparing to remove the four IP addresses that do not belong there.

Iterate through the remove list

I built an iterative statement to prepare to remove elements of the remove list from the **ip_addresses** list:

```
# 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:

  # 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:

    # Display `element` in every iteration

    print(element)
```

To do this, I built a **for** loop to iterate through **ip_addresses** and used **element** as the loop variable and **in** as the loop condition.

Remove IP addresses that are on the remove list

To remove the IP addresses from the **ip_addresses** list that are on the remove list, I built on the previous code using a conditional statement:

```
# 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:

  # 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)

# Display `ip_addresses`

print(ip_addresses)
```

To build a conditional statement, I used the **if** condition and the **in** operator to see if an element is in **remove_list**. If elements in **remove_list** were also in **ip_addresses**, I used the **.remove()** function to remove them.

To see the updated ip_addresses list, I used the **print()** function and found that the four IP addresses in the remove_list were removed from the allow list.

Update the file with the revised list of IP addresses

Finally, I updated the original file that was used to create the **ip_addresses** list. To do this, I used the **.join()** function to convert ip_addresses back in a string format to rewrite the file.The .join() method takes the list that I converted from a string and concatenates every element into a string.

In the beginning, I used the **with** keyword, **import_file** variable, the "**r**" parameter, and the **.read()** function to read the contents of the **allow_list.txt** file. This time, I applied the "**w**" parameter and the **.write()** function to change the file to the update **ip_addresses** list.

```
# 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:

  # 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:

  # Rewrite the file, replacing its contents with `ip_addresses`

  text = file.write(ip_addresses)
```

Summary

[Add content here.]