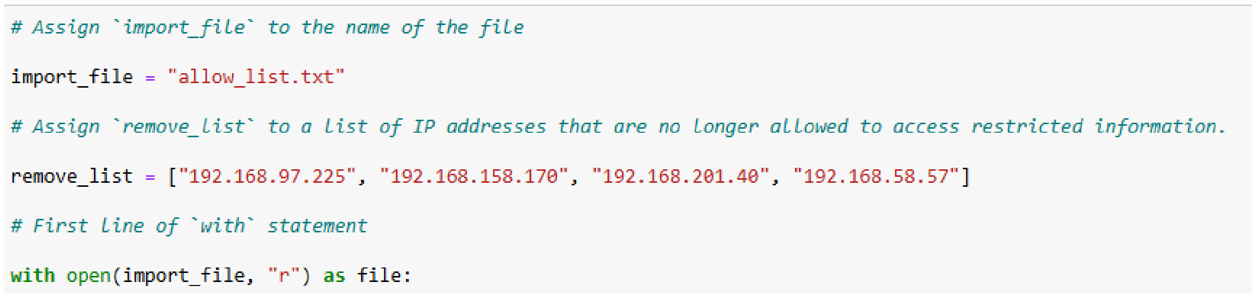
Algorithm for file updates in Python

Project description

You are a security professional working at a health care company. As part of your job, you're 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 access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

Your 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. If so, you should remove those IP addresses from the file containing the allow list.

Open the file that contains the allow list



```python
# 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:
```

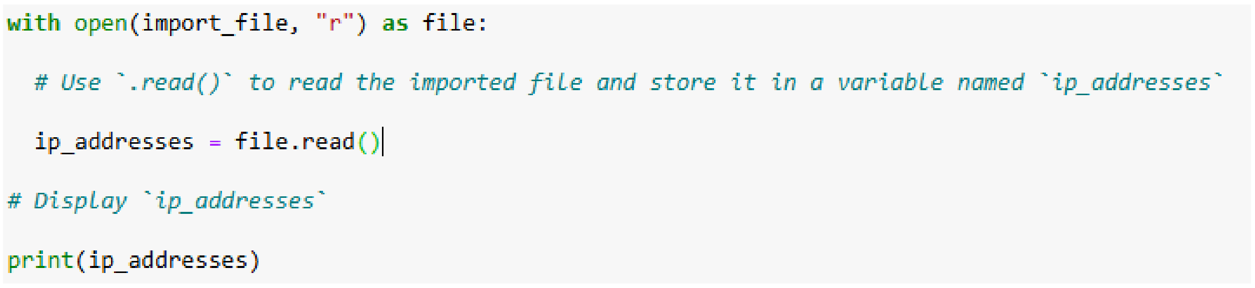
In my algorithm, I use the `with` statement and the `open()` function in read mode to open the allow list file so I can read the IP addresses in it. The `with` keyword helps by automatically closing the file after the code inside the `with` block is done.

In `with open(import_file, "r") as file:`, the `open()` function has two inputs:

1. The name of the file to open.
2. `"r"` to say I want to read the file.

The `as` keyword creates a variable called `file`, which holds the file's content while I work in the `with` block.

Read the file contents

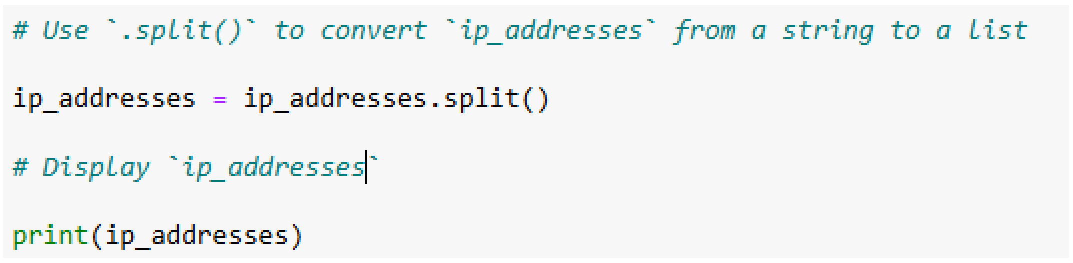


```
with open(import_file, "r") as file:

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

  ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

When I use the `open()` function with the argument `"r"` for "read," I can use the `.read()` method inside the `with` block. The `.read()` method turns the file into a string so I can work with it. I used `.read()` on the `file` variable from the `with` statement and saved the result as a string in a variable called `ip_addresses.`

In short, this code reads the contents of "allow_list.txt" and saves it as a string so I can later organize and extract data in my Python program.

Convert the string into a list



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

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

The `.split()` function is used by adding it to a string variable. It turns the string into a list. Splitting `ip_addresses` into a list makes it easier to remove IP addresses from the allow list. By default, `.split()` separates the text at spaces and puts each piece into a list.

In this algorithm, the `.split()` function takes the string of IP addresses stored in `ip_addresses`, where each IP is separated by a space, and converts it into a list of IP addresses. I then saved this list back into the `ip_addresses` variable.

Iterate through the remove list

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

for element in ip_addresses:
```

The for loop in Python repeats code for each item in a sequence. In a Python program like this, the for loop applies specific code to each item in a list. The for keyword starts the loop, followed by the loop variable element and the word in. The word in tells the loop to go through the ip_addresses list and give each item to the element variable.

Remove IP addresses that are on the remove list

```
# 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)
```

Inside my for loop, I made a check to see if the element was in the ip_addresses list. I did this to avoid errors, because using .remove() on something not in the list would cause a problem. Then, inside the check, I used .remove() on ip_addresses and passed in the element so that each IP in the remove_list would be taken out of ip_addresses.

Update the file with the revised list of IP addresses

```
  # 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)

# Call `update_file()` and pass in "allow_list.txt" and a list of IP addresses to be removed

update_file('allow_list.txt', ["192.168.25.60", "192.168.140.81", "192.168.203.198"])

# Build `with` statement to read in the updated file

with open("allow_list.txt", "r") as file:

  # Read in the updated file and store the contents in `text`

  text = file.read()
```

The `.join()` method puts all items in a list into one string. It's applied to a string that tells how to separate the items when they're joined. In this program, I used `.join()` to turn the `ip_addresses` list into a string so I could use it as an argument for the `.write()` method when saving to the "allow_list.txt" file.

I used the `with` statement to rewrite the original file and replace its contents.

After that, I called the `update_file()` function, passing the "allow_list.txt" file along with the IP addresses that needed to be removed as parameters.

Summary

I created an algorithm to remove IP addresses from the "allow_list.txt" file that are listed in the `remove_list` variable. The algorithm opens the file, converts it into a string, and then turns that string into a list stored in the `ip_addresses` variable. I then went through each IP address in the `remove_list`. For each one, I checked if it was in the `ip_addresses` list, and if it was, I used the `.remove()` method to delete it. Finally, I used the `.join()` method to turn the updated `ip_addresses` list back into a string and overwrite the "allow_list.txt" file with the new list of IP addresses.