ALGORITHM FOR FILE UPDATES IN PYTHON

Project description

As a security professional at a healthcare company, my job is to regularly update files to make sure only authorized employees have access to confidential patient information. Employees have access to content based on their level of authorization, and whether their IP address is on the allow or remove list. I created an algorithm to automate updating the file and remove IP addresses that no longer have access.

Open the file that contains the allow 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"]
# First line of `with` statement
with open("allow_list.txt", "r") as file:
```

Used the **with** statement, which helps manage files and the **open()** function to open the “allow_list.txt” text file. The function takes in two parameters. The first is the file to import, and the second parameter “r” indicates that I want to read the file. The **as** statement assigns a variable named file, which stores the output of the **open()** function.

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:
  # 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)
```

After opening the file, I used the **read()** function to convert the file into a string which allows me to read it. Then, I assigned the output of this method to `ip_addresses`.

Convert the string into a 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()

# Display `ip_addresses`

print(ip_addresses)
```

In order to remove individual IP addresses from the allow list, I need to convert the string into a list. I used the **split()** function and assigned the output back to the same variable ip_addresses.

Iterate through the remove 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 `remove_list`

for element in remove_list:

  # Display `element` in every iteration

    print(element)
```

A key part of my algorithm involves iterating through the remove list. To do this, I incorporated a **for** loop, which has the purpose of applying specific code statements to all elements in a sequence. The keyword **in** indicates to iterate through the sequence ip_addresses and assign each value to the loop variable element.

Remove IP addresses that are on the remove 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 `remove_list`

for element in remove_list:

    # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

        # use the `.remove()` method to remove
        # elements from `ip_addresses`

        ip_addresses.remove(element)

# Display `ip_addresses`

print(ip_addresses)
```

My algorithm requires removing any IP addresses from the allow list that are also on the remove list. Within my **for** loop, I created a conditional statement that evaluated whether the loop variable element was found in the ip_addresses list. Within that conditional, I used the **.remove()** function and passed in the element variable so that each IP address that was in the remove_list would be removed from ip_addresses.

Update the file with the revised list of IP addresses

```
# 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 `remove_list`

for element in remove_list:

    # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

        # use the `.remove()` method to remove
        # elements 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 = "\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)
```

As a final step in my algorithm, I needed to update the allow list with the revised list of IP addresses. To do so, I first used the **.join()** function to convert the list back into a string. This was necessary in order to pass the string ip_addresses as an argument while writing the file. I used the "\n" string to instruct Python to place each element on a new line. Then, I used another **with** statement to update the file. Instead of opening the file to read it, this time I want to open the file to write over it indicated by the "w". When using the "w" argument, I can call the **.write()** function and append it to the object file that I identified in the **with** statement. Finally, I passed in the ip_addresses variable to specify what to replace the original file with.

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 into a string to be read, and then converting the string into a list stored in the variable ip_addresses. Then, I iterated through the IP addresses in the remove_list and removed the IP addresses that no longer had access with the .remove() method. Finally, I used the .join() method to convert the list back into a string in order to write over the contents of the file with the revised list of IP addresses.