Algoritmo para atualizações de arquivos em Python

Descrição do projeto:

Em minha organização, o acesso a conteúdo restrito é controlado com uma lista de permissão de endereços IP. O arquivo "allow_list.txt" identifica esses endereços IP. Uma lista de remoção separada, identifica endereços IP que não devem mais ter acesso a esse conteúdo. Criei um algoritmo para automatizar a atualização do arquivo "allow_list.txt" e remover esses endereços IP que não devem mais ter acesso.

Abra o arquivo que contém a lista de permissões:

Para a primeira parte do algoritmo, abri o "allow_list.txt" . Primeiro, atribuí o nome como uma string para a variável import_file:

```
# Atribua 'arquivo de importação' ao nome do arquivo
import_file = "allow_list.txt"
```

Então, usei uma instrução ‘with’ para abrir:

```
#Construir a instrução 'with' para ler o conteúdo inicial do arquivo
with open(import_file, "r") as file:
```

No meu algoritmo, a declaração ‘with’ é usada com a função ‘.open()’ no modo de leitura para abrir o arquivo da lista de permissões com o propósito de lê-lo. O propósito de abrir o arquivo é me permitir acessar os endereços IP armazenados no arquivo da lista de permissões. A palavra-chave ‘with’ ajuda a gerenciar os recursos fechando o arquivo após sair da

declaração ‘with’. No código com open(import_file, "r") as file:, a função ‘open()’ tem dois parâmetros. O primeiro identifica o arquivo a ser importado e o segundo indica o que eu quero fazer com o arquivo. Neste caso, "r" indica que eu quero lê-lo. O código também usa a palavra-chave ‘as’ para atribuir uma variável chamada file; file armazena a saída da função .open() enquanto eu trabalho dentro da declaração ‘with’.

Ler o conteúdo do arquivo:

Para ler o conteúdo do arquivo, usei o método .read() para convertê-lo em string.

```
#Use 'read.()' para ler o arquivo importado e armazená-lo em uma variável
chamada 'IP_addresses'
ip_addresses = file.read()
```

Ao usar uma função .open() que inclui o argumento "r" para “ler”, posso chamar o Função .read() no corpo da instrução ‘with’. O método .read() converte em uma string e me permite lê-lo. Apliquei o método .read() à variável de arquivo identidade e na instrução ‘with’. Em seguida, atribuí a saída de string deste método a variável ip_addresses.

Em resumo, este código lê o conteúdo do "allow_list.txt" arquivo em um formato de string, Isso me permite usar posteriormente a string para organizar e extrair dados em meu código Python.

Converter a string em uma lista:

Para remover endereços IP individuais da lista de permissões, eu precisava que estivessem no formato de lista. Portanto, em seguida usei o método .split() para converter a string ip_addresses em uma lista:

```
#Use 'split()' para converter 'ip_addresses' de uma string em uma lista
ip_addresses = ip_addresses.split()
```

A função .split() é chamada anexando-a a uma variável de string. Funciona convertendo o conteúdo de uma string para uma lista. O propósito da divisão inserir ip_addresses em uma lista é torná-lo mais fácil remover endereços IP da lista de permissões. Por padrão, a função .split() divide o texto por espaço em branco em elementos da lista. Neste algoritmo, a função .split() pega os dados armazenados na variável ip_addresses, que é uma sequência de

endereços IP separados por um espaço em branco e converte essas strings em uma lista de endereços IP. Para armazenar esta lista, eu retribuo de volta à variável ip_addresses.

Iterar pela lista de remoção:

Uma parte importante do meu algoritmo envolve a interação pelos endereços IP que são elementos da lista_remover. Para fazer isso, incorporei um loop for:

```
#Crie um patrimônio interativo
#Variável de loop de nome 'elemento'
#Loop Percorrer 'remove_list'
for element in remove_list:
```

O loop for em Python repete o código para uma sequência específica. O propósito geral do loop for em um algoritmo Python como este é aplicar instruções de código específicas a todos os elementos em uma sequência. A palavra-chave ‘for’ inicia o loop for. Ela é seguida pela variável de loop element e pela palavra-chave ‘in’. A palavra-chave ‘in’ indica para iterar pela sequência ip_addresses e atribuir cada valor à variável de loop element.

Remover endereços IP que estão na lista de remoção.

Meu algoritmo requer a remoção de qualquer endereço IP da lista de permissões, ip_addresses, que também é contido em remove_list. Como não havia duplicatas em ip_addresses, eu estava capaz de usar o seguinte código para fazer isso:

```
for element in remove_list:

#Crie uma instrução condicional para avaliar se 'element' está em
'ip_addresses'

    if element in ip_addresses:

#Use o método '.remove()' para remover
```

```
#Elementos de 'ip_addresses'

        ip_addresses.remove(element)
```

Primeiro, dentro do meu loop for, criei uma condicional que avaliou se a variável do loop elemento foi encontrado na lista ip_addresses. Eu fiz isso porque aplicar .remove() a elementos que não foram encontrados em ip_addresses resultam em erro.

Então, dentro dessa condicional, apliquei .remove() a ip_addresses. eu passei no loop elemento variável como argumento para que cada endereço IP que estava na remove_list seria removido de ip_addresses.

Atualizar o arquivo com a lista revisada de endereços IP.

Como etapa final do meu algoritmo, precisei atualizar o arquivo de lista de permissão com a lista revisada de endereços IP. Para fazer isso, primeiro precisei converter a lista de volta para uma string. Usei o método .join() para isso:

```
#Converter 'endereços IP' de volta para uma string para que possa ser escrito no arquivo de texto
ip_addresses = "\n".join(ip_addresses)
```

O método .join() combina todos os itens em um iterável em uma string. O método .join() é aplicado a uma string contendo caracteres que separam os elementos no iterável uma vez unidos em uma string. Neste algoritmo, usei o método .join() para criar uma string da lista ip_addresses para que eu pudesse passá-la como um argumento para o método .write() ao escrever no arquivo "allow_list.txt". Usei a string ("\n") como separador para instruir o Python a colocar cada elemento em uma nova linha.
Então, usei outra declaração ‘with’ e o método ‘.write()’ para atualizar o arquivo.

```
#Construir declaração 'with' para reescrever o arquivo original
with open(import_file, "w") as file:
#Reescreva o arquivo, substituindo seu conteúdo por 'ip_addresses'
    File.write(ip_addresses)
```

Desta vez, usei um segundo argumento de "w" com a função open() na minha declaração ‘with’. Este argumento indica que eu quero abrir um arquivo para escrever sobre seu conteúdo. Ao usar este argumento "w", eu posso chamar a função .write() no corpo da declaração ‘with’. A função .write() escreve dados de string em um arquivo especificado e substitui qualquer conteúdo de arquivo existente.

Neste caso, eu queria escrever a lista de permissões atualizada como uma string para o arquivo "allow_list.txt". Dessa forma, o conteúdo restrito não será mais acessível a nenhum endereço IP que foi removido da lista de permissões. Para reescrever o arquivo, eu anexei a função .write() ao arquivo de objeto do arquivo que eu identifiquei na declaração ‘with’. Eu passei a variável ip_addresses como o argumento para especificar que o conteúdo do arquivo especificado na declaração ‘with’ deve ser substituído pelos dados nesta variável.

Resumo:

Criei um algoritmo que remove endereços IP identificados, em uma variável remove_list do arquivo "allow_list.txt" de endereços IP aprovados. Esse algoritmo envolve abrir o arquivo, convertê-lo em uma string para ser lida e, em seguida, converter essa string em uma lista armazenada na variável ip_addresses. Em seguida, iterei pelos endereços IP em remove_list. A cada iteração, foi avaliado se o elemento fazia parte da lista ip_addresses. Se fizesse, aplicaria o método.remove(), para remover o elemento de ip_addresses. Depois disso, usei o método .join() para converter os ip_addresses de volta em uma string para que eu pudesse sobrescrever o conteúdo do arquivo "allow_list.txt" com a lista revisada de endereços IP.