Atualizando um arquivo usando Python

Por Egon Fernandes

Descrição do projeto

Na minha organização, o acesso a conteúdos restritos é controlado por uma lista de IPs permitidos. O arquivo "allow_list.txt" guarda esses endereços. Existe também uma lista de remoção, que mostra quais IPs não devem mais ter o acesso. Eu criei um código para automatizar a atualização do "allow_list.txt" e tirar esses IPs que não devem mais ter acesso.

Abrindo um arquivo que contém a “allow list”

Na primeira parte do código, eu abri o arquivo "allow_list.txt". Primeiro, coloquei o nome desse arquivo como uma string na variável import_file:

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

import_file = "allow_list.txt"
```

Depois, eu usei um comando “with” para abrir o arquivo:

```
# Build `with` statement to read in the initial contents of the file

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

No meu código, o comando “with” é usado junto com a função “open()” no modo de leitura para abrir o arquivo da lista de permissões. A ideia de abrir esse arquivo é poder acessar os endereços IP que estão guardados lá. O “with” ajuda a gerenciar os recursos porque fecha o arquivo automaticamente quando saímos do bloco. No código “with open(import_file, "r") as file:”, a função “open()” tem dois parâmetros: o primeiro é o arquivo que quero abrir (que está na variável “import_file”), e o segundo indica o que vou fazer com ele. Nesse caso, o "r" mostra que quero apenas ler. O código também usa o “as” para criar a variável “file”, que guarda o resultado da função “open()” enquanto estiver dentro do bloco do “with”.

Ler o conteúdo do arquivo

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

```
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()
```

Quando uso a função “open()” com o argumento "r" de leitura, eu posso chamar a função “.read()” dentro do bloco do “with”. O método “.read()” transforma o arquivo em uma string e me permite ler seu conteúdo. Apliquei o “.read()” na variável “file” definida no “with”, e depois guardei o resultado (a string) na variável “ip_addresses”.

Resumindo, esse código lê o conteúdo do arquivo "allow_list.txt" em formato de string, o que me permite depois organizar e extrair dados no meu programa em Python.

Converter a string em uma lista

Para conseguir remover endereços IP da lista de permissões, eu precisava que ela estivesse no formato de lista. Então usei o método “.split()” para converter a string “ip_addresses” em uma lista:

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

ip_addresses = ip_addresses.split()
```

A função “.split()” é chamada anexada a uma variável do tipo string. Ela transforma o conteúdo da string em uma lista. A ideia de dividir “ip_addresses” em lista é facilitar a remoção dos IPs. Por padrão, o “.split()” separa o texto pelos espaços em branco. Nesse código, o “.split()” pega os dados da variável “ip_addresses” (uma string com IPs separados por espaços) e converte em uma lista de endereços IP. Para guardar essa lista, atribuí de volta à variável “ip_addresses”.

Iterar pela lista de remoção

Uma parte importante do código é percorrer os endereços IP que estão dentro da “remove_list”. Para isso, usei um loop “for”:

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

for element in remove_list:
```

O loop “for” no Python repete um trecho de código para cada item de uma sequência. A ideia geral do “for” nesse algoritmo é aplicar um mesmo código a todos os elementos de uma lista. A palavra-chave “for” inicia o loop, em seguida da variável “element” e da palavra “in”. O “in” indica que vamos percorrer a “lista ip_addresses” e, a cada volta, atribuir um valor à variável “element”.

Remover endereços IP que estão na lista de remoção

O algoritmo precisa remover qualquer endereço IP da lista “ip_addresses” que também esteja na “remove_list.” Como não havia duplicados em “ip_addresses”, consegui fazer isso com o seguinte código:

```
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)
```

Primeiro, dentro do “for”, criei uma condição para verificar se o elemento estava na lista “ip_addresses”. Fiz isso porque tentar usar “.remove()” em algo que não existe na lista daria erro.
Depois, dentro dessa condição, usei “.remove()” em “ip_addresses”. Passei o “element” como argumento, assim cada IP da “remove_list” era retirado de “ip_addresses”.

Atualizar o arquivo com a lista revisada de endereços IP

Na etapa final do código, precisei atualizar o arquivo da lista de permissões com a lista revisada de IPs. Para isso, primeiro tive que converter a lista de volta em string. Usei o método “.join()” para isso:

```
# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = "\n".join(ip_addresses)
```

O método “.join()” junta todos os itens de uma lista em uma string. Ele é chamado em uma string que define o que vai separar os elementos. Nesse código, usei o “.join()” para transformar a lista “ip_addresses” em string, e depois passei isso como argumento para o “.write()” na hora de escrever no arquivo "allow_list.txt". Usei o separador "\n" para que cada item ficasse em uma linha diferente.

Depois, usei outro “with” junto com o método “.write()” para atualizar o arquivo:

```
# 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)
```

Dessa vez, usei o argumento "w" com a função “open()” no “with”. Esse argumento indica que quero abrir o arquivo para sobrescrever o conteúdo. Quando uso "w", posso chamar o “.write()” dentro do bloco do “with”. O “.write()” escreve a string no arquivo e substitui tudo o que já estava lá.

Nesse caso, eu queria escrever a lista revisada de IPs no arquivo "allow_list.txt". Assim, o conteúdo restrito não ficaria mais acessível para os IPs removidos da lista. Para sobrescrever o arquivo, usei “.write()” no objeto file definido no “with” e passei a variável “ip_addresses” como argumento, substituindo o conteúdo do arquivo pelos dados dessa variável.

Em Resumo

Eu criei um programa que remove da lista de permissões ("allow_list.txt") os endereços IP que aparecem na variável “remove_list”. Esse algoritmo abre o arquivo, converte em string para leitura e depois transforma em lista armazenada em “ip_addresses”. Em seguida, percorre os IPs de “remove_list” e, a cada volta, verifica se o IP está em “ip_addresses”. Se estiver, usa o “.remove()” para tirar da lista. Depois, usei o “.join()” para transformar “ip_addresses” de volta em string e sobrescrever o arquivo "allow_list.txt" com a lista revisada.