Algorithm For File Updates in Python

Project description

As a security professional working at a health care company, I must regularly update a file that identifies the employees who can access restricted content. Employees have restricted access based on their IP address. Also, there is a remove list that identifies which employees must be removed from this allow list. Therefore, I created an algorithm in Python to automatize this task to check and remove any IP address identified on the remove list.

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

Assign a variable name to the name of the file. This will help to read better the code and reuse it.

The open() function in Python allows to open a file.

As the first parameter, it takes in the name of the file (or a variable containing the name of the file). As the second parameter, it takes in a string that indicates how the file should be handled.

Pass in the letter "r" as the second parameter when you want to read the file.

Read the file contents

```
# 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()
```

The .read() method in Python allows you to read in a file.

Call file.read() to read the imported file.

To display the contents of a variable, pass it as an argument to the print() function.

Convert the string into a list

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

ip_addresses = ip_addresses.split()
```

The `.split()` method in Python allows you to convert a string to a list. This method can take in a parameter that specifies which character to split on. If a parameter is not passed in, the method will split on whitespace by default. Note that whitespace includes any space between text on the same line and the space between one line and the next line.

Each IP address is on a new line in the `allow_list.txt` file. In other words, there is whitespace between IP addresses in the text file. When you use `.split()`, it will separate the IP addresses and output them as a list.

Iterate through the remove list

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

for element in remove_list:
```

Build a `for` loop to iterate through `remove_list`. It must start with the `for` keyword. Here I use `element` as the loop variable and use `in` as the loop condition.

Remove IP addresses that are on the 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)
```

To remove element from ip_addresses, call the .remove() method on ip_addresses, and pass in element.

To remove element from ip_addresses, call ip_addresses.remove() and pass in element.

Applying the .remove() method in this way is possible because there are no duplicates in the ip_addresses list.

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 = "\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)
```

The .join() method takes in an iterable (such as a list) and concatenates every element of it into a string. The .join() method is applied to a string consisting of the character that will be used to separate every element in the iterable once its converted into a string. In the code below, the method is applied to the string "\n". The "\n" character indicates to separate each element by placing it on a new line. The argument of the .join() method is the iterable you want to convert, and in this case, that's ip_addresses. As a result, it converts ip_addresses from a list back into a string with each IP address on a new line.

To complete the first line of the with statement, call the open() function and pass in the name of the file as the first parameter and the letter "w" as the second parameter.

The "w" parameter specifies that you're opening the file for the purpose of writing to it.

Inside the with statement, call the .write() method to replace the contents of the file with the data stored in ip_addresses.

Inside the with statement, call file.write() and pass in ip_addresses.

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 to a string to be read, and then converting this string to a list stored in the variable ip_addresses. I then iterated through the IP addresses in remove_list. With each iteration, I evaluated if the element was part of the ip_addresses list. If it was, I applied the .remove() method to it to remove the element from ip_addresses.. After this, I used the .join() method to convert the ip_addresses back into a string so that I could write over the contents of the "allow_list.txt" file with the revised list of IP addresses.