Brijen Jayeshbhai Shah

Update a file through a Python algorithm

Project description

Our organization uses an "allow list" system to control access to restricted content. IP addresses are stored in a file called "allow_list.txt" and any address that shouldn't have access is added to a separate "remove list." I developed an algorithm to automate the update process, ensuring only authorized IP addresses have access.

Open the file that contains the allow list

For the first part of the algorithm, I opened the "allow_list.txt" file. First, I assigned this file name as a string to the import_file variable:

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

import_file = "allow_list.txt"
```

Then, I used a with statement to open the file:

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

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

My code uses a *with statement* to automatically open and read the "allow list" file containing authorized IP addresses. This ensures the file is properly closed after use. The *open function* takes two arguments: the file path and the access mode ("r" for reading). A variable named *file* is assigned to the opened file for easy access within the *with* block.

Read the file contents

In order to read the file contents, I used the .read() method to convert it into the 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()
```

The code reads the *"allow_list.txt"* file into a string variable for easier manipulation within the Python program. It achieves this using the *read()* method, which extracts the file's contents as a string. This string is then stored in the *ip_addresses* variable, ready for further processing.

Convert the string into a list

In order to remove individual IP addresses from the allow list, I needed it to be in list format. Therefore, I next used the .split() method to convert the ip_addresses string into a list:

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

ip_addresses = ip_addresses.split()
```

The code transforms the string of IP addresses into a list using the *split()* method. This list structure makes it more convenient to remove specific IP addresses later in the algorithm. The *split()* method, by default, separates string elements based on whitespace, resulting in a list where each item represents a single IP address. The updated list is then stored back into the *ip_addresses* variable for further processing.

Iterate through the remove list

I used *for loop* for iterations.

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

for element in remove_list:
```

The code employs a *for loop* to tackle each IP address in the *ip_addresses* list systematically. This loop ensures instructions within its block are executed for every individual IP address in the list. The keyword *for* initiates the loop, followed by the loop variable *element*. The keyword *in* designates that the loop will iterate through the *ip_addresses* list, assigning each IP address to the *element* variable during each cycle.

Remove IP addresses that are on the remove list

My algorithm requires removing any IP address from the allow list, ip_addresses, that is also contained in remove_list. Because there were not any duplicates in ip_addresses, I was able to use the following code to do this:

```
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)
```

Within the for loop, a conditional check takes place to ensure each IP address is present in the ip_addresses list before attempting removal. This precautionary step prevents potential errors that could arise from attempting to remove non-existent entries. If the IP address is confirmed as present, the remove() method is employed to remove it from the list. The loop variable element, containing the current IP address being processed, is passed as an argument to the remove() method, guaranteeing accurate removal of addresses intended for deletion.

Update the file with the revised list of IP addresses

As a final step in my algorithm, I needed to update the allow list file with the revised list of IP addresses. To do so, I first needed to convert the list back into a string. I used the .join() method for this:

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

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

The .join() method combines all items in an iterable into a string. The .join() method is applied to a string containing characters that will separate the elements in the iterable once joined into a string. In this algorithm, I used the .join() method to create a string from the list ip_addresses so that I could pass it in as an argument to the .write() method when writing to the file "allow_list.txt". I used the string ("\n") as the separator to instruct Python to place each element on a new line.

Then, I used another with statement and the .write() method to update the file:

```
# 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)
```

This time, I used a second argument of "w" with the open() function in my with statement. This argument indicates that I want to open a file to write over its contents. When using this argument "w", I can call the .write() function in the body of the with statement. The .write() function writes string data to a specified file and replaces any existing file content.

In this case I wanted to write the updated allow list as a string to the file "allow_list.txt". This way, the restricted content will no longer be accessible to any IP addresses that were removed from the allow list. To rewrite the file, I appended the .write() function to the file object file that I identified in the with statement. I passed in the ip_addresses variable as the argument to specify that the contents of the file specified in the with statement should be replaced with the data in this variable.

Summary

I designed a Python algorithm that automates the removal of specific IP addresses from an "*allow list*" file. The algorithm first opens the "allow list" file in a read-only mode to access its contents. It then converts the file's IP address string into a modifiable list for easier manipulation. Moreover, The algorithm cycles through each IP address in a designated "*remove list*." It checks if any of these IP addresses exist within the "*allow list*" and selectively removes them using the remove() method. Afterwards, The algorithm converts the updated "allow list" back into a string format using the join() method. Finally, it overwrites the original "allow list" file with the revised list of IP addresses, ensuring only authorized addresses maintain access.