**Updating a File Through a Python Algorithm**

**Project description**

In my organization, I manage access to restricted content through an allow list of IP addresses. The file 'allow\_list.txt' contains these IP addresses, while a separate remove list identifies IP addresses that should no longer have access. I created an algorithm to automatically update the 'allow\_list.txt' file by removing these IP addresses.

**Steps**

**1. Open the File That Contains the Allow List**

In the first step of the algorithm, I opened the 'allow\_list.txt' file. I began by assigning the file name as a string to the import\_file variable. Next, I used a **with** statement to open the file. The with statement and the .open() function in read mode were used to access the IP addresses in the allow list file. The **with** statement helped manage resources, ensuring the file was closed once it was no longer needed. The open() function had two parameters: the first specified the file, and the second ('r') indicated the file was opened for reading.

**2. Read the File Contents**

To read the file contents, I applied the .read() method to convert the file into a string. When the file was opened in read mode ('r'), the .read() method converted the file contents into a string, allowing me to read it. I used the .read() method on the variable 'file' and assigned the string output to the variable 'ip\_addresses.' This process allowed me to access and manipulate the IP addresses stored in 'allow\_list.txt.'

3. **Convert the String Into a List**

To remove specific IP addresses from the allow list, I first needed to convert the string into a list. I used the .split() method for this. The .split() method converted the contents of the string into a list by splitting it by whitespace, creating list elements for each item in the string. By doing this, I converted the string of IP addresses into a list, making it easier to remove specific addresses.

**4. Iterate Through the Remove List**

A key part of my algorithm involved iterating over the IP addresses in the remove list. To achieve this, I used a for loop. The for loop allowed Python to repeat code for a specific sequence. The loop checked each IP address, applying the relevant code to each one in turn.

**5. Remove IP Addresses That Are on the Remove List**

The goal of the algorithm was to remove any IP addresses from 'ip\_addresses' that were also present in the remove list. Since there were no duplicates in 'ip\_addresses,' I created a conditional statement within the for loop to check if the current loop variable ('element') was found in 'ip\_addresses.' If the element was present, I applied the .remove() method to remove it from the list.

**6. Update the File with the Revised List of IP Addresses**

The final step was to update the 'allow\_list.txt' file with the revised list of IP addresses. To do this, I converted the list back into a string using the .join() method. The .join() method combined all elements in the list into a string. I used a newline character ('\n') as the separator, which placed each IP address on a new line in the file. After converting the list into a string, I used another with statement to open the file in write mode ('w') and applied the .write() method to overwrite the file's contents.

**Error Handling:**

To ensure the reliability of my algorithm, I implemented error handling for cases where the 'allow\_list.txt' or 'remove\_list.txt' files might not exist or be inaccessible. I used a try-except block to catch and handle any exceptions, ensuring the program could provide informative feedback and prevent unexpected crashes.

**Optimization and Scalability:**

I optimized the algorithm by storing IP addresses in a set rather than a list, which reduced the time complexity for lookups. This optimization allowed the algorithm to scale more effectively when processing larger datasets, significantly improving performance with minimal resource usage.

**Testing and Validation**

To ensure accuracy, I thoroughly tested the algorithm using sample IP lists. I created mock versions of the 'allow\_list.txt' and 'remove\_list.txt' files, compared the output against expected results, and validated that the correct IP addresses were removed.

**Future Enhancements**

I will add logging functionality in future iterations to track which IP addresses were removed from the allow list. Additionally, I could automate the script to run periodically or on-demand, ensuring the 'allow\_list.txt' file is continuously updated with the latest security policies.

**Version Control**

Throughout development, I used Git for version control to manage the algorithm's iterations. This allowed me to track changes, revert to previous versions, and collaborate effectively with team members.

**Summary**

The algorithm I developed successfully removes IP addresses from the 'allow\_list.txt' file if found in the remove list. This process involved opening the file, reading its contents, converting them into a list, and iterating through the remove list to remove any matching IP addresses. Finally, I converted the list back into a string and wrote the updated IP addresses to the file.