Algorithm for file updates in Python

Project description

This project is an exercise in demonstrating the use of algorithms and automation for the cybersecurity professional. This is the scenario as outlined in the project: " You are a security professional working at a healt care company. As part of your job, you're required to regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.
Your task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, you should remove those IP addresses from the file containing the allow list."

Open the file that contains the allow list

The first step in constructing an algorithm in Python is to import any necessary files. In this exercise you are given a list of allowed IP addresses as well as a list of addresses that need to be removed from the allowed list. The `with` statement and open() function are how this is accomplished.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

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

Read the file contents

Using the `.read` function within the body of the `with` statement allows the file to be read and converts it into a string in Python.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

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

    ip_addresses = file.read()

print(ip_addresses)
```

Convert the string into a list

The `.split` function converts the string into a list. A string is immutable and constant and cannot be modified after it is created. Because of this it has to be changed into a list in order to add or modify data within the file.

```
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
with open(import_file, "r") as file:
    ip_addresses = file.read()
ip_addresses = ip_addresses.split()
print(ip_addresses)
```

Iterate through the remove list

Iteration of the remove list using a conditional `for` statement instructs Python to parse through the data and analyze each IP address to determine if it meets the condition requiring its removal.

```
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
with open(import_file, "r") as file:
    ip_addresses = file.read()
ip_addresses = ip_addresses.split()
for element in ip_addresses:
    print(ip_addresses)
```

Remove IP addresses that are on the remove list

If any of the IP addresses in the remove list are found in the import file Python removes them from the list. The `.remove(element)` function is used to accomplish this task.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

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

    ip_addresses = file.read()

ip_addresses = ip_addresses.split()

for element in ip_addresses:

    if element in remove_list:

        ip_addresses.remove(element)

print(ip_addresses)
```

Update the file with the revised list of IP addresses

FInally the list is revised to include only the list of allowed IP addresses after the addresses no longer allowed are removed. The `.join` method is used to return the file to a string before overwriting the existing file with the newly updated list.

```
import_file = "allow_list.txt"

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

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

    ip_addresses = file.read()

ip_addresses = ip_addresses.split()

for element in ip_addresses:

    if element in remove_list:

        ip_addresses.remove(element)

ip_addresses = " ".join(ip_addresses)

with open(import_file, "w") as file:

    file.write(ip_addresses)
```

Summary

This project demonstrates my ability to use Python to automate and analyze data in the role of a security professional. IP addresses, usernames and passwords, internet and network protocols, and many other types of data are a regular part of the security profession. It is necessary to eliminate as many repetitive and tedious tasks as possible to allow the analyst to focus on making sense of the data and applying it to the context of security risks, threats, and vulnerabilities. In today's age of information, the more work that can be allocated to machines and computing devices, the more the analyst is able to give attention to the human element. Python is an excellent tool and extremely well suited to the task of automation as demonstrated in the above exercise.