# Update A File Through A Python Algorithm

## Project description

As a security professional at a healthcare company, I was tasked with managing access to a restricted subnetwork by updating an allow list of IP addresses. These IPs represent employees permitted to access personal patient records. I developed a Python algorithm that reads from this allow list, compares it to a separate remove list of unauthorized IPs, and updates the file by removing any matches. This helps maintain secure access control efficiently and accurately.

————————————————————————————————————————

## Open the file that contains the allow list

To open the file, I used a `with` statement and the built-in `open()` function with the "r" (read) mode:

```python

import\_file = "allow\_list.txt"

with open(import\_file, "r") as file:

```

This ensures that the file is safely opened and automatically closed after use.

————————————————————————————————————————

## Read the file contents

Inside the `with` block, I used the `.read()` method to convert the contents of the allow list into a string and assigned it to a variable:

```python

ip\_addresses = file.read()

```

This allows easy string manipulation in subsequent steps.

————————————————————————————————————————

## Convert the string into a list

To make each IP address individually accessible, I used the `.split()` method to convert the string into a list:

```python

ip\_addresses = ip\_addresses.split()

```

This is necessary because list operations (like removing items) are not possible on strings.

————————————————————————————————————————

## Iterate through the remove list

I used a `for` loop to iterate through the `remove\_list` with `element` as the loop variable:

```python

for element in remove\_list:

```

This allowed me to check each element against the current IPs in the allow list.

————————————————————————————————————————

## Remove IP addresses that are on the remove list

Inside the loop, I included a conditional that checks if the current `element` is in the `ip\_addresses` list:

```python

if element in ip\_addresses:

ip\_addresses.remove(element)

```

I used `.remove()` because it deletes the first occurrence of the specified element. This is safe here because the IP list contains no duplicates.

————————————————————————————————————————

## Update the file with the revised list of IP addresses

To write the updated IPs back into the file, I first joined the list into a string using `.join()`:

```python

ip\_addresses = "\n".join(ip\_addresses)

```

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

```python

with open(import\_file, "w") as file:

file.write(ip\_addresses)

```

This completes the algorithm by saving the changes to the allow list.

————————————————————————————————————————

## Summary

This Python algorithm opens a file containing an allow list of IP addresses, reads and parses its contents, and compares it against a remove list. It then removes any disallowed IPs, formats the data back into string format, and updates the original file. It uses built-in functions such as `open()`, `.read()`, `.split()`, `.remove()`, `.join()`, and `.write()` for efficient file handling. The `with` statement ensures that files are properly opened and closed, preventing resource leaks. This script automates a critical part of access management, enhancing both security and efficiency.