# Writing an Algorithm For File Updates in Python

## Project description:

While working as security professionals at our healthcare organization, it is our duty to ensure that access to restricted content is regulated while maintaining the availability of information for approved network access.

The `allow_list.txt` file identifies the IP addresses of devices that are permitted access to restricted content while a separate `remove_list` identifies which IP addresses should no longer have access.

For example, only devices from approved hospital networks or VPN connections may access patient records. When a remote clinic loses authorization for any reason, their IP addresses are added to the remove list and automatically blocked from the system.

In this project, I created an algorithm that automatically updates the `allow_list.txt` file by removing any IP addresses that are identified to be on the `remove_list`.

## Open the file containing 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:
```

The `open()` function in Python allows me to open a file; the first parameter takes in the name of the file (`import_file`) and the second parameter takes in a string which indicates how the file should be handled. I input `"r"` for the second parameter thereby telling Python that I wish to read the file:

## Reading the file's contents:

```
# 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"]

# 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()

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.3
7 192.168.156.224 192.168.60.153 192.168.69.116
```

In order to read the file, I call the `.read()` method on the file and assign its output to a variable that I named, `ip_addresses`. I then simply use the `print()` function on the newly named created variable in order to display the contents of the file.

To summarize, this code simply reads the contents of the `allow_list.txt` file and converts it into a single large string of IP addresses.

## Converting the String into a List:

In [7]:
```
# 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"]

# 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()

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

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.1
68.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

Next, I need to separate the IP addresses inside the string so that I can manipulate them individually without affecting the rest. The best way to achieve this is by converting this string into a list using the `.split()` method. This method can accept a parameter that specifies which

character to split on. Without providing a parameter, the method will split on whitespace by default (whitespace includes any space between text on the same line and the space between one line and the next line).

Since each IP address is on a new line in the `allow_list.txt` file, there are whitespaces between each of them; this now results in a list of IP addresses that are separated by commas and each have their own index.

## Iterating through the Remove List:

```
In [9]: # 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"]

        # 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()

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

        ip_addresses = ip_addresses.split()

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

        for element in ip_addresses:

            # Display `element` in every iteration

            print(element)

        ip_address
        192.168.205.12
        192.168.6.9
        192.168.52.90
        192.168.90.124
        192.168.186.176
        192.168.133.188
        192.168.218.219
        192.168.52.37
        192.168.156.224
        192.168.60.153
        192.168.69.116
```

I created a `for` loop to iterate through each index/element inside the `ip_addresses` list. I set `element` as the loop variable and `in` as the loop condition. This loop now iterates through each IP address stored inside the `ip_addresses` list and prints each of them using the `print()` function.

## Removing IP addresses that are on the Remove List:

```
In [8]: # 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"]

        # 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()

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

        ip_addresses = ip_addresses.split()

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

        for element in ip_addresses:

          # Build conditional statement
          # If current element is in `remove_list`,

            if element in remove_list:

                # then current element should be removed from `ip_addresses`

                ip_addresses.remove(element)

        # Display `ip_addresses`

        print(ip_addresses)

        ['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.16
        8.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

In order to remove IP addresses from our `allow_list` list I once again iterate through each element inside of the `ip_addresses` list and I check to see if the current `element` matches an element stored inside of `remove_lists`; this is accomplished using an `if` statement that compares the two elements as iteration continues.

When there is a successful match on an element, the `.remove()` method is called on the `ip_addresses` list to remove that ip address (`element`) from the `ip_addresses` list.

## Updating the File with the Revised List of IP addresses:

In [10]:
```
# 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"]

# 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()

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

ip_addresses = ip_addresses.split()

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

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

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

ip_addresses = " ".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 it's converted into a string. In the code below, the method is applied to the string `" "`, which contains just a space character. The argument of the `.join()` method is the iterable I wish 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 a space between each element and the next.

To complete the first line of the `with` statement, I 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 it is 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:

In [ ]:

```
# Define a function named `update_file` that takes in two parameters: `import_file` and `remove_list`
# and combines the steps you've written in this lab leading up to this

def update_file(import_file, remove_list):

  # 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()

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

  ip_addresses = ip_addresses.split()

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

  for element in ip_addresses:

    # Build conditional statement
    # If current element is in `remove_list`,

    if element in remove_list:

      # then current element should be removed from `ip_addresses`

      ip_addresses.remove(element)

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

  ip_addresses = " ".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)

# Call `update_file()` and pass in "allow_list.txt" and a list of IP addresses to be removed

update_file("allow_list.txt", ["192.168.25.60", "192.168.140.81", "192.168.203.198"])

# Build `with` statement to read in the updated file

with open("allow_list.txt", "r") as file:

  # Read in the updated file and store the contents in `text`

  text = file.read()

# Display the contents of `text`

print(text)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.3
7 192.168.156.224 192.168.60.153 192.168.69.116
```

Pictured above is the code cell of a complete algorithmic function that updates a list of authorized IP addresses that are stored inside a .txt file:
The function receives two arguments:

The first argument accepts a file that contains a list of authorized IP addresses and the second argument accepts a list of unauthorized IP addresses that need to be removed.
This function will check whether any unauthorized IP addresses are found within the authorized list. If any matches are found, the function will then remove them from the authorized list. Finally, the function writes the newly modified list of authorized IP addresses back onto the original file.