

# Algorithm for file updates in Python

## Project description

As a cybersecurity analyst at a health care company, I am responsible for controlling employee access to a restricted subnetwork that holds personal patient data. Access is granted based on IP addresses listed in a file named `allow_list.txt`. I created a Python algorithm to update this file by removing IP addresses that appear on a separate `remove_list`. This process ensures only authorized users can connect to sensitive systems, reducing security risks.

## Open the file that contains the allow list

The `with` statement automatically handles closing the file after the code block executes. The `open()` function is used with the `"r"` mode to read the contents of the file.

```
In [2]: # 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 inform

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:

File "<ipython-input-2-b925af1022fc>", line 11
    with open(import_file, "r") as file:
        ^
SyntaxError: unexpected EOF while parsing
```

## Read the file contents

This string stores all IP addresses as one block of text, which we will later convert into a list to allow filtering and modification.

```

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

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

    ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)

```

```

ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116

```

## Convert the string into a list

The `.split()` method divides the string into a list of IP addresses, separated by whitespace (typically newline characters).

```

In [4]: # 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("\n")

# Display `ip_addresses`

print(ip_addresses)

```

```

['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90',
'192.168.158.170', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.16
8.201.40', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57',
'192.168.69.116', '']

```

## Iterate through the remove list

This loop visits each IP address in the `remove_list` one at a time.

```
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
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

## Remove IP addresses that are on the remove list

This method works as expected here because the IP addresses are unique, there are no duplicates in the list.

```
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', '192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116' ]

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

`.join()` combines the list into a formatted string.

`with open(import_file, "w")` opens the file in write mode.

`.write()` replaces the file content with the updated IP addresses.

```
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)

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

    # Rewrite the file, replacing its contents with `ip_addresses`

    file.write(ip_addresses)
```

## Summary

This Python algorithm automates the process of maintaining access control based on IP addresses in a healthcare organization. The script opens a file, reads its contents, splits the data into a list, removes unauthorized entries based on a second list and writes the revised list back to the file. Key features of the algorithm include the use of the with statement for safe file handling, string to list conversion with `.split()`, and list modification using `.remove()`. The final result is a secure, accurate list of IPs with access to sensitive systems.