Algorithm for File Updates in Python

Project Description

The purpose of this algorithm is to demonstrate methodology for controlling access to restricted content. It provides a function for maintaining a file that contains a list of IP addresses that are allowed to access restricted content.

Breakout of Algorithm Components

Open the file that contains the allow list:

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

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

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

Read the file contents:

```
  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

    ip_addresses = file.read()
```

Convert the string into a list:

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list:

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

for element in ip_addresses:
```

Remove IP addresses that are on the remove list:

```
    if element in remove_list:

        # then current element should be removed from `ip_addresses`

 ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses:

```
# 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)
```

Summary

While the components that makeup this algorithm are simple, the sum of their parts, when implemented via a function, demonstrates the strength of this design approach when managing large amounts of log data, as it encourages reuse and improves maintainability of enhancements and eases debugging overhead, as the developer does not need to contend with many, potentially repetitive, blocks of code.

Application and Output:

In [9]:

```
# 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 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.5
2.37 192.168.156.224 192.168.60.153 192.168.69.116
```