Algorithm for file updates in Python

Project description

In this cybersecurity project, the objective is to create an algorithm that helps manage access to restricted content by parsing a file containing authorized IP addresses. As a security analyst, you will automate the process of reading, updating, and maintaining the list of IP addresses. The algorithm will specifically remove IP addresses that no longer have permission to access the restricted information, ensuring that only authorized users can reach sensitive content. This project demonstrates key concepts in access control, file parsing, and automation using Python.

Open the file that contains 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:
```

Read the file contents

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

```
  ip_addresses = file.read()

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

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

ip_addresses = ip_addresses.split()

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', '192.16
8.90.124', '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']
```

Iterate through the remove list

```
# 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

```
  # 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', '192.1
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']
```

Update the file with the revised list of IP addresses

```
# 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)
```

Summary

In this project, I implemented an algorithm to manage access control for restricted content by working with a text file containing a list of allowed IP addresses. I began by using the **`read()`** function to read the entire file's contents into a string, which was then converted into a list of IP addresses using **`.split()`**. This split the string at each space, creating a list where each element was a distinct IP address.

Next, I iterated through the list of IP addresses and compared each address to a **`remove_list`** that contained IPs no longer allowed access. Using a conditional **`if`** statement, I removed any matching IPs from the list. After processing, I joined the updated list back into a string using **`.join()`**, so it could be written back to the file, replacing the original content.

Concepts involved included file reading and writing with the **`with`** statement, string and list manipulation using **`read()`**, **`.split()`**, and **`.join()`**, and iterating with conditional removal logic.