Algorithm for file updates in Python

Project description

In this project, I developed an algorithm to manage access control by updating a text file containing IP addresses. This file lists the addresses allowed to access restricted content within our organization. My task was to automate the process of removing IP addresses that no longer have access, ensuring only authorized addresses remain.

Open the file that contains the allow list

I used a `with` statement to open the file named `import_file` in read mode. This approach ensures that the file is properly closed after reading, which is crucial for maintaining file integrity and preventing resource leaks.

Read the file contents

To read the contents of the file, I utilized the `.read()` method. This allowed me to store all IP addresses as a single string in a variable called `ip_addresses`.

Convert the string into a list

I converted the string of IP addresses into a list using the `.split()` method. This step was necessary to facilitate iteration and manipulation of individual IP addresses.

Iterate through the remove list

I implemented a loop to iterate over each element in `ip_addresses`. By using a conditional statement, I checked if each IP address was present in `remove_list`.

Remove IP addresses that are on the remove list

If an IP address was found in `remove_list`, I removed it from `ip_addresses`. This ensured that only authorized IP addresses remained in the list.

Update the file with the revised list of IP addresses

After processing, I converted the list back into a string using `" ".join(ip_addresses)`. Then, I opened the file again in write mode and replaced its contents with this updated string. This step finalized the update process by saving changes directly to the file.

Summary

Through this project, I successfully automated the management of access control lists using Python. By developing an algorithm that reads, processes, and updates IP address data, I streamlined a critical cybersecurity task. This approach not only enhances security but also improves efficiency by reducing manual intervention.

```
# Define a function named `update_file` that takes in two parameters:
`import_file` and `remove_list`

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)
```