Algorithm for file updates in Python

Project description

As a security professional at a healthcare company, I manage access to restricted records using an **allow list (`allow_list.txt`)** that contains approved IP addresses. When access is revoked, those IPs must be removed.

I developed a Python algorithm to automate this process, ensuring security and efficiency. The algorithm:

1. Opens and reads the allow list file
2. Converts the data into a list
3. Identifies and removes unauthorized IP addresses
4. Writes the updated list back to the file

This automation enhances security by preventing unauthorized access and maintaining an accurate allow list.

1. Open the file that contains the allow list

I started by opening the allow list file and reading its contents. Using a `with` statement ensures the file is properly managed and closed after reading.

```
# Assign filename to a variable
import_file = "allow_list.txt"

# Open the file in read mode
with open(import_file, "r") as file:
    ip_addresses = file.read()
```

What I Did:

- Used `open()` with the `"r"` parameter to read the file.
- Used `with` to automatically close the file.
- Stored the contents in `ip_addresses` as a string.

2. Read the file contents

Next, I ensured that the file contents were stored in a variable for further processing.

```
# Read the file contents into a variable
with open(import_file, "r") as file:
    ip_addresses = file.read()
```

What I Did:

- Used `.read()` to retrieve the file's content as a string.
- Stored the data in `ip_addresses` for further processing.

3. Convert the string into a list

Since each entry needed to be processed separately, I converted the string into a list using `.split()`.

```
# Convert the string into a list of IP addresses
ip_addresses = ip_addresses.split()
```

What I Did:

- Used `.split()` to convert the string into a list.
- Ensured each entry is stored as an individual element.

4. Iterate through the remove list

I created a `remove_list` containing the entries that needed to be removed and iterated through it.

```
# Iterate through the remove list and remove matching entries
for entry in remove_list:
    if entry in ip_addresses:
        ip_addresses.remove(entry)
```

What I Did:

- Used a `for` loop to check if each entry in `remove_list` existed in `ip_addresses`.

- Used `.remove(entry)` to delete unauthorized records.

5. Remove IP addresses that are on the remove list

After removing the entries, I converted the list back into a string format before writing it to the file.

```
# Convert the updated list back into a string
ip_addresses = "\n".join(ip_addresses)

# Write the updated list back to the file
with open(import_file, "w") as file:
    file.write(ip_addresses)
```

What I Did:

- Used `.join()` to merge the updated list into a string.
- Opened the file in `"w"` mode to overwrite the existing content.
- Used `.write()` to save the revised allow list.

6. Update the file with the revised list of IP addresses

To make this process more efficient, I created a function that performs all steps at once.

```
# Define a function to update the allow list
def update_file(import_file, remove_list):
    with open(import_file, "r") as file:
        ip_addresses = file.read().split()

    ip_addresses = [entry for entry in ip_addresses if entry not in remove_list]

    with open(import_file, "w") as file:
        file.write("\n".join(ip_addresses))
```

Why I Used a Function:

- Modular: I can reuse this function for different files.
- Efficient: It consolidates all steps into one.
- Scalable: I can expand it without rewriting the entire script.

Summary

This project automates the removal of unauthorized entries from the allow list using Python. The steps I followed include:

1. Opening and reading the file using `with open()`.
2. Converting the file content into a list using `.split()`.
3. Iterating through the removal list to identify revoked entries.
4. Eliminating unauthorized records using `.remove()`.
5. Writing the revised list back with `.join()` and `.write()`.
6. Encapsulating the process in a function for efficiency and reusability.

By automating this process, I improved security and reduced manual errors, ensuring accurate access control.