Algorithm for file updates in Python

Project description

In this project, I developed a Python algorithm to automate the update of an allow list file used by a healthcare company to control employee access to restricted content. The allow list (stored in a file named `allow_list.txt`) contains IP addresses permitted to access a secure subnetwork, while a separate remove list identifies IPs that should be taken off this allow list. This algorithm reads the file, converts its content into a list, removes any IP addresses found in the remove list, and then updates the file. This solution reduces manual work, minimizes errors, and helps ensure that only authorized IP addresses remain active.

Open the file that contains the allow list

1. Opening and Reading the File

To start, I assigned the file name to a variable and used a **with statement** combined with the `open()` function to read the file:

python
Copy

```
import_file = "allow_list.txt"
with open(import_file, "r") as file:
    ip_addresses_str = file.read()
```

Explanation:

- The **with statement** ensures the file is properly closed after reading.
- The `open()` function is used with the `"r"` mode (read mode).
- The `.read()` method converts the file's contents into a string, stored in `ip_addresses_str`.

Convert the string into a list

Next, I converted the string into a list of IP addresses using the **.split()** method:

python
Copy
```
ip_addresses = ip_addresses_str.split("\n")
```

Explanation:

- The `.split("\n")` method splits the string at each newline, resulting in a list of IP addresses.

Iterate through the remove list

I defined a list of IP addresses that need to be removed and used a **for loop** with an **if statement** to remove each matching IP:

python
Copy
```
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40",
"192.168.58.57"]

for ip in remove_list:
    if ip in ip_addresses:
        ip_addresses.remove(ip)
```

Explanation:

- The **for loop** iterates through each element in `remove_list`.
- The **if statement** checks whether the current element exists in the `ip_addresses` list.
- The **.remove()** method removes the IP address if found. This works effectively because the allow list does not contain duplicate entries.

Update the file with the revised list of IP addresses

After removing the unwanted IPs, I joined the list back into a string and wrote the updated data back to the file:

python
Copy
```
updated_ip_addresses = "\n".join(ip_addresses)

with open(import_file, "w") as file:
    file.write(updated_ip_addresses)
```

Explanation:

- The **.join()** method with `"\n"` as the separator converts the list back into a newline-separated string.

- The file is then reopened in write mode (`"w"`) using a **with statement**, and the `.write()` method updates the file with the new string.

Completed Code

```
# Define the file to be updated
import_file = "allow_list.txt"

# Open the file and read its contents as a string
with open(import_file, "r") as file:
    ip_addresses_str = file.read()

# Convert the string into a list of IP addresses (split by newline)
ip_addresses = ip_addresses_str.split("\n")

# Define the remove list of IP addresses to be removed
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40",
"192.168.58.57"]

# Iterate through the remove list and remove matching IP addresses
from the allow list
for ip in remove_list:
```

```
    if ip in ip_addresses:
        ip_addresses.remove(ip)

# Convert the updated list back into a string with newline separators
updated_ip_addresses = "\n".join(ip_addresses)

# Open the file in write mode and update it with the revised list
with open(import_file, "w") as file:
    file.write(updated_ip_addresses)
```

Summary

This algorithm demonstrates key Python file operations essential for cybersecurity automation. I used the **with statement** with the `open()` function to ensure proper file handling. The **.read()** and **.write()** methods were used for file input/output, while **.split()** converted file content into a list for processing. The **for loop** combined with an **if statement** and the **.remove()** method enabled efficient removal of unauthorized IP addresses. Finally, the **.join()** method was used to prepare the updated data for file rewriting. This approach streamlines updating security configurations, reducing manual intervention and error, which is critical in managing dynamic network environments.