Algorithm for file updates in Python

Project description

In this project, I developed a Python algorithm to manage and update an IP address allow list. The algorithm reads the contents of an existing file, converts the data into a manageable format, and removes specific IP addresses based on a provided list of addresses to be excluded. This process ensures that only approved IP addresses remain in the allow list. The final step involves updating the original file with the revised list, demonstrating effective file handling and data manipulation in Python.

Open the file that contains the allow list

```
import_file = "allow_list.txt"

with open(import_file) as file:
    # Code to read and parse the file content goes here
```

`import_file`: A variable that stores the name of the file you want to open.
`with`: A context manager that ensures the file is properly closed after its contents are read or processed.
`open()`: A built-in Python function that opens a file and returns a file object.
`as`: A keyword that assigns the opened file object to a variable (in this case, `file`).

Read the file contents

```
import_file = "allow_list.txt"

with open(import_file) as file:
    ip_addresses = file.read()
```

`file.read()`: This method reads the entire content of the file as a single string. If the file is large, be aware that this method will load all data into memory.
`ip_addresses`: A variable that stores the string containing the entire contents of the **`allow_list.txt`** file.

Convert the string into a list

```
import_file = "allow_list.txt"

with open(import_file) as file:
    ip_addresses = file.read()

ip_list = ip_addresses.split()
```

`ip_addresses.split()`: This method splits the `ip_addresses` string into a list of IP addresses based on the specified delimiter (default is any whitespace).
`ip_list`: A variable that stores the list of IP addresses after splitting the string.

Iterate through the remove list

```
for element in remove_list:
   # Code to remove element from ip_list goes here
```

`for element in remove_list:` This line initiates a loop that iterates through each item in the `remove_list`. The variable `element` represents the current item in the list during each iteration.

Remove IP addresses that are on the remove list

```
for element in remove_list:
   if element in ip_list:
      ip_list.remove(element)
```

`remove_list` is present in the `ip_list`.
`ip_list.remove(element)`: This method removes the first occurrence of `element` from `ip_list`. Since there are no duplicates in `ip_list`, this approach successfully removes the IP address.

Update the file with the revised list of IP addresses

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

# 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)
```

`"\n".join(ip_list)`: This expression joins the elements of `ip_list` into a single string, with each IP address separated by a newline character (`\n`).
`open(import_file, 'w')`: Opens the file assigned to `import_file` in write mode (`'w'`), which allows you to overwrite the file's contents.
`file.write(updated_ip_addresses)`: Writes the `updated_ip_addresses` string (containing the revised list of IP addresses) to the file.

Summary

The algorithm begins by opening and reading the contents of a file named `allow_list.txt`, storing the data as a string. This string is then split into a list of IP addresses, which can be easily managed and modified. The algorithm iterates through a separate list of IP addresses that need to be removed and checks if each address exists in the allow list. If found, the IP address is removed. After updating the list, the algorithm converts it back into a string format, with each IP address placed on a new line. Finally, the revised list is written back to the original file, effectively updating the allow list to reflect the changes.