Algorithm for file updates in Python

Project description

In this scenario, I work as a security professional within a healthcare company, where it is necessary to safeguard patient data. My responsibility involves maintaining a file that specifies which employees have access to personal patient records. Access control is enforced based on the employees' IP addresses.

I developed a Python algorithm that automates the process of checking whether the IP addresses in the allow list match with any IP addresses listed in the remove list. If there is a match, the algorithm removes those IP addresses from the allow list. This process ensures that only authorized personnel have access to patient records, maintaining the confidentiality and security of sensitive healthcare information.

Open the file that contains the allow list

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Open file with `with` statement

with open (import_file, "r") as file:
```

a. Create a variable named 'import_file' and assign the string value "allow_list.txt" to it.
b. Use 'with' statement to handle the file safely.
c. Use the 'open()' function to open the file specified in the 'import_file' variable. The 'r' argument indicates that the file is being opened in read mode.
d. Use 'as file' to assign the opened file object to the variable file.

Read the file contents

```
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()
```

e. Use the '.read()' method to read the entire content of the file and return it as a string, which is then stored in the variable 'ip_addresses'.

Convert the string into a list

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

ip_addresses = ip_addresses.split()
```

f. Use the '.split()' method to convert the 'ip_addresses' string into a list format.

Iterate through the remove list

```
# Build iterative statement
# Name loop variable `element`
# Loop through `remove_list`

for element in remove_list:
```

g. Set up a 'for' loop that iterates through each element in the 'remove_list' list. During each iteration, the loop variable 'element' takes the value of the current element in 'remove_list'.

Remove IP addresses that are on the remove list

```
for element in 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)
```

h. If the current element from the 'remove_list' is present in the 'ip_addresses' list, the '.remove()' method is used to remove that element.
i. The '.remove()' method can be used here because there are no duplicates in the ip_addresses list.

Update the file with the revised list of IP addresses

```python
# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = "\n".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)
```

j. Use '.join' to join the elements of 'ip_addresses' into a single string and "\n" to place each IP address on a new line
k. Open the 'import_file' in write mode ('w') using a 'with' statement to ensure that the file is properly closed after writing the updated content.
l. Use '.write' to write the updated IP address string back to the file.

Summary

The developed algorithm addresses the critical task of managing access control for a healthcare company's sensitive data. The process begins by reading the existing IP addresses from the "allow_list.txt" file into a Python list. Next, the algorithm iterates through a remove_list containing IP addresses that need to be revoked. For each element in the remove_list, the algorithm checks and removes the corresponding IP address from the original list. After removing the specified IP addresses, the updated list is converted back into a string, with IP addresses separated by newline characters. Using the with statement and the .write() method, the algorithm then overwrites the content of the "allow_list.txt" file with the revised IP addresses, ensuring that the file reflects the current access permissions accurately. This process makes certain that only authorized employees have access to restricted healthcare data, which enhances security and maintains regulatory compliance.