Algorithm for file updates in Python

Project description

A health care organization needs me to regularly update a file of employees who can access restricted access level content. The file includes employee IP addresses that currently have access to personal patient records, and I have been given a list of IP addresses to remove. In this project, I create a Python algorithm that checks whether the allow list contains any of the IP addresses to remove, and remove them.

Open the file that contains the allow list

I first start by storing the allow_list.txt file path to a variable that I can use to open it.

```
import_file = "allow_list.txt"
```

Then using the *with open() as file* syntax I open the file to read the currently allowed IP addresses listed in the file.

```
with open(import_file) as file:
```

Read the file contents

With the file open, I can now read the contents with the *.read()* method which converts the contents of the file into a python string.

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

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to a

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.5

# Build `with` statement to read in the initial contents of the file

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

Convert the string into a list

Since I will need to iterate through the list of IP addresses to remove any unallowed IP addresses, I convert the file contents string into a Python list with *.split()*.

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list

Now that all of the allowed IP addresses were easily available in a list, I iterated through all of the IP addresses I needed to remove so that I could check if they existed in the currently allowed IP addresses.

```
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", '
```

```
for element in remove_list:
```

Remove IP addresses that are on the remove list

While iterating through the IP addresses to remove, I created a conditional to remove an IP address from the list of allowed IP addresses if it was currently allowed but should be removed.

```
if element in ip_addresses:

  # use the `.remove()` method to remove
  # elements from `ip_addresses`

    ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses

Now that I had the updated list of IP addresses, I needed to update the allow_list.txt file with the updated list. I first converted the Python list back to a string where each IP address is on its own line. Then I opened the allow_list.txt file with the *"w"* parameter to completely overwrite the file to avoid duplicate information. Finally, I wrote the updated string of allowed IP addresses to the file.

```
# 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)
```

Summary

For this task, I created a Python algorithm that takes in a file of IP addresses and uses a list of IP addresses to remove to update the file. Using Python functionalities like *with open*, *.read()*, and *.remove()* I convert the file to a Python list and iterate through the list of IP addresses to remove and compare them to the IP addresses currently listed in the file. After removing the invalid IP addresses, I convert the list back to a string to then update the valid IP addresses file with the new list.