Algorithm for file updates in Python

Project description

This project required developing an algorithm that parses a file containing IP addresses allowed to access restricted content and remove IP addresses that no longer have access rights.

Open the file that contains the allow list

I used the **import_file** variable to encompass the allow list file (allow_list.txt) and used a **with** statement to open it.

```
# 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 access restricted ir

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# First line of `with` statement

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

Read the file contents

I used the **.read()** method to convert the imported file to a readable string and stored it in a variable named **ip_addresses.**

```
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()
```

Convert the string into a list

I reassigned the **ip_addresses** variable so its data type is updated from a string to a list using the **.split()** function. This makes removing IP addresses much more feasible compared to when the **ip_addresses** variable was in string form.

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

ip_addresses = ip_addresses.split()
```

Iterate through the remove list

Next, I built the iterative statement that is capable of removing the IP addresses in the **remove_list** from the **ip_addresses** list using a **for** loop with the variable called **element.**

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

for element in ip_addresses:
```

Remove IP addresses that are on the remove list

Next, I expanded on the previous step by adding a conditional that checked if the variable **element** was found in the **ip_addresses** list and the **remove_list**, and if it was, that element i.e., IP address, was removed from the list of allowed IP addresses via the **.remove** function.

```
for element in ip_addresses:

  # 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)
```

Update the file with the revised list of IP addresses

Lastly, I updated the allow list after the unauthorized IP addresses were removed. First, using **.join()** to return the **ip_addresses** variable back into a string, which then allowed for **.write()** to be used to overwrite/update the **“allow_list.txt”** file.

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

ip_addresses = " ".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)
```