Jasper Davis

Algorithm for File Updates in Python

Project description

This project showcases how Python can be used to take a text file containing IP addresses

Open the file that contains the allow list

First, the name of the text file is saved as a variable called `import_file`.

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

import_file = "data/allow_list.txt"
```

Then, a `with` statement is used to open the file:

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

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

Read the file contents

The opened file is saved to a variable called `ip_addresses`.

```
  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

    ip_addresses = file.read()
```

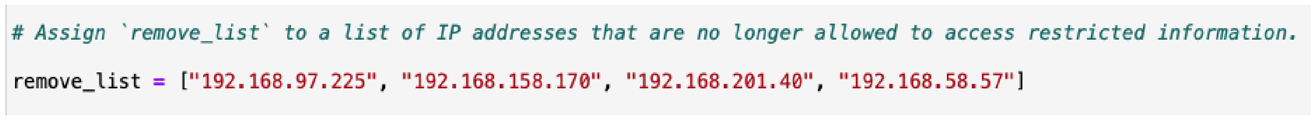
Convert the string into a list

The `.split()` function is called on the `ip_addresses` variable and then saved back into that same variable. This converts the single string of IP addresses into a list of strings, one for each IP address.

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

ip_addresses = ip_addresses.split()
```

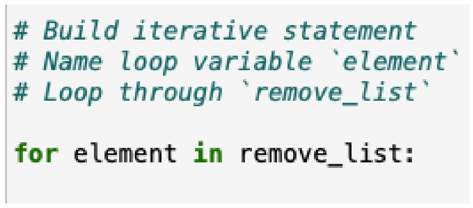
Iterate through the remove list

Earlier in the program, we created a variable named `remove_list`, which represented a list of IP addresses that should be removed from the initial `allow_list` text file.



```
# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

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

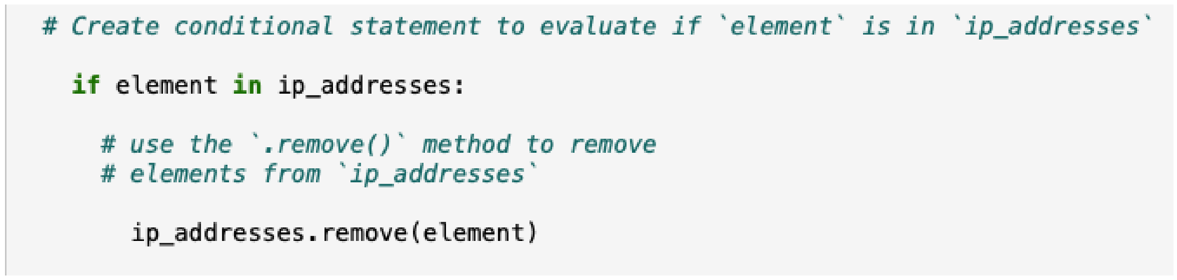
We then use a `for` loop to iterate through each element (a string, in this case) in the `remove_list`.



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

for element in remove_list:
```

Remove IP addresses that are on the remove list

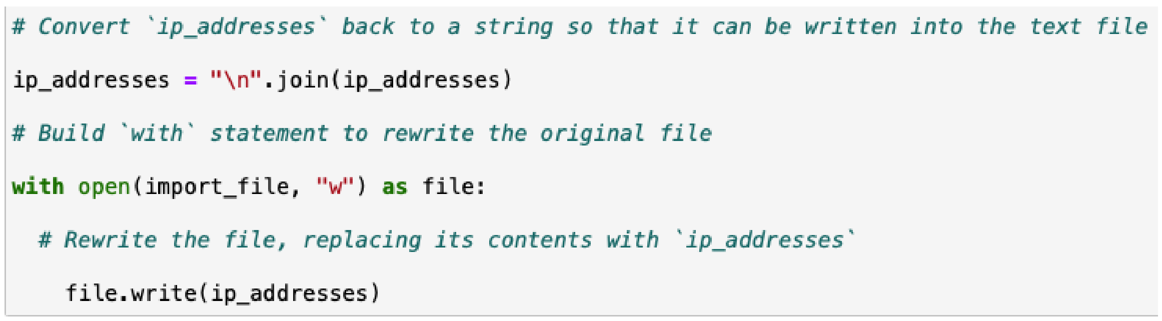
Using an `if` statement, we compare each string in the `remove_list` to each element in the `ip_addresses` list. If an element is present, it is then removed from the `ip_addresses` list.



```
  # Create conditional statement to evaluate if `element` is in `ip_addresses`

    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

At the end, we then take our `ip_addresses` list and convert it back into a single string with each IP address separated by `/n` (the character that represents a newline). We then open our `import_file` once more, in writing (“`w`”) mode, using a `with` statement. We conclude by writing our `ip_addresses` string to the opened text file.



```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)
```

Summary

Using Python, a simple algorithm was created that would take specified IP addresses (`remove_list`), compare them to an existing whitelist of allowed IP addresses (`"allow_list.txt"`), and remove those specified.