Algorithm for file updates in Python

Project description

In this scenario, I am a security analyst at a healthcare company, and am responsible for regularly updating a file that identifies employees that can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is also an allow list for IP addresses permitted access to the restricted subnetwork. There is also a remove list with a list of IP addresses to be removed from the allow list. I have been tasked with creating an algorithm to check whether the allow list contains any IP addresses identified on the remove list; if so, the IP address(es) is to be removed.

Open the file that contains the allow list

In [1]:
```
# 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 information.
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:
```

We first opened the file that contained the allow list with the “r” parameter, because we will be reading the contents first. This is opened as a variable named “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()

# Display `ip_addresses`

print(ip_addresses)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

We then read the imported file using .read() and stored it in a variable named “ip_addresses”, and displayed the results.

Convert the string into a list

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

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.170', '192.16
8.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.52.37', '192.
168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

In order to remove IP addresses, we need to convert the string to a list. To do this, we used .split() on ip_addresses, and then displayed the results. When no parameter is passed, each element is separated with whitespace.

Iterate through the remove list

```
for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

A second list named remove_list contains a list of IP addresses to be removed from the ip_addresses list. We need to build an iterative statement to loop through the list. We created a loop variable, “element”, to iterate through ip_addresses and list them individually. We then displayed these results.

Remove IP addresses that are on the remove list

```
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)

# Display `ip_addresses`

print(ip_addresses)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.16
8.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

We included code that would remove IP addresses that are on the remove list and allow list. First, a conditional statement was created. The conditional statement evaluates if the “element” variable is in ip_addresses, and remove_list, then that specific element should be removed from ip_addresses using .remove(). Once this conditional was complete, the result was printed.

Update the file with the revised list of IP addresses

```
# 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(ip_addresses, "w") as file:
  # Rewrite the file, replacing its contents with `ip_addresses`
  file.write(ip_addresses)
```

Lastly, after removing the correct IP addresses from the ip_addresses variable, we needed to update the file with the revised list of IP addresses. However, we could not do this in list form. In order to convert this list back to string form, we applied the method .join() to “\n” in order to separate the returned elements on new lines. Next, we created another “with” statement, using “w” as a parameter in order to rewrite the file with the new text. We then applied .write() to file, with the updated ip_addresses being the parameter.

Summary

This algorithm takes in an imported list and remove_list, reads an imported file using the .read() method, and assigns this to a variable named ip_addresses. Because we are unable to remove elements from strings, we need to convert the string of IP addresses to a list. In order to do this, the .split() method is applied. Once ip_addresses is converted to a list, we are able to build an iterative statement and loop through ip_addresses, using a “for” loop. If a loop variable, “element”, is in ip_addresses, and also in remove_list, then that element variable is to be removed. Once that’s complete, the list is converted back into a string so that the text file can be updated or rewritten. We convert the list back to a string by applying the .join() method on “\n”. We then use another “with” statement to open the file with the “w”, since we will be replacing the text file with the updated “ip_addresses”. Storing this as a function allows this to be called multiple times from anywhere.