Algorithm for file updates in Python

Project description

In this activity I created an algorithm that parses a series of IP addresses that can access restricted information and removes the addresses that are no longer allowed.

Open the file that contains the allow list

The "`allow_list.txt`" file contains a series of IP addresses that are allowed to access restricted information.

The "`remove_list`" variable contains the list of IP addresses to be removed.

I used the "`print()`" function to display the contents of the "`allow_list.txt`" file and "`remove_list`" variable.

```
In [2]: # 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"]

        # Display `import_file`

        print(import_file)

        # Display `remove_list`

        print(remove_list)
```

```
allow_list.txt
['192.168.97.225', '192.168.158.170', '192.168.201.40', '192.168.58.57']
```

Read the file contents

Using the `.read()` method allowed me to read the imported file and store it in a variable named `ip_addresses`.

Afterwards, displayed `ip_addresses` to examine the data in its current format.

In [8]:
```
# 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"]

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

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
```

Convert the string into a list

I used the "`.split()`" method to convert a string to a list. Each IP address is on a new line in the "`allow_list.txt`". The "`.split()`" method it will separate the IP addresses and output them as a list.

```
In [9]: # 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"]

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

        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()

        # 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', '192.168.158.17
0', '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']
```

Iterate through the remove list

First, I built a `for` loop to iterate through `ip_addresses`. Named the loop variable `element` and used `in` as the loop condition.
Second, to display the contents of the variable, I passed it as an argument to the `print()` function.

```
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
```

Remove IP addresses that are on the remove list

To remove the elements of `remove_list` from the `ip_addresses` list, I built a conditional statement in an iterative statement that loops through `ip_addresses`.

Afterwards, to remove `element` from `ip_addresses`, I called `ip_addresses.remove()` and pass in `element`.

```
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', '192.168.186.17
6', '192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153',
'192.168.69.116']
```

Update the file with the revised list of IP addresses

To update the original file that was used to create the `ip_addresses` list, I added a line of code containing the `.join()` method that takes in an iterable, in this case the `ip_addresses` list, and concatenates every element of it into a string.

After this line with the `.join()` method, I built the `with` statement that rewrites the original file. Inside the `with` statement, I called `file.write()` and passed in `ip_addresses` to replace the contents of the file with the data stored in `ip_addresses`.

```
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)

# 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)
```

Summary

Using Python to create an algorithm, helped me to automate the task that I was assigned to do.

I created an algorithm that removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of approved IP addresses. This algorithm involved opening the file, converting it to a string to be read, and then converting this string to a list stored in the variable `ip_addresses`. I then iterated through the IP addresses in `remove_list`. With each iteration, I evaluated if the element was part of the `ip_addresses` list. If it was, I applied the `.remove()` method to it to remove the element from `ip_addresses`.. After this, I used the `.join()` method to convert the `ip_addresses` back into a string so that I could write over the contents of the `"allow_list.txt"` file with the revised list of IP addresses.