Algorithm for file updates in Python

Project description

Create an algorithm using Python to check an allow list against a remove list and remove IP's that should be removed.

Open the file that contains the allow list

First, we will open the file through Python using:

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

This tells Python that we will be pulling the data from import_file variable (which is representing allow_list.txt)

Read the file contents

To read the contents of the file we will use add another variable indented under the 'with statement'

```
  ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

This sets the variable ip_addresses as a read function for the imported file. So when we print ip_addresses, we're telling Python to print the contents of the import_file variable.

```
192.168.150.224
192.168.60.153
192.168.58.57
192.168.69.116
```

Convert the string into a list

Converting the resulting string to a list is simple, we will be using the .split function to turn the string into a list format.

```
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)
```

By simply adding .split() with no arguments, we are telling it to use whitespaces as the break point between entries.

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.1
70', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.21
9', '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

To iterate through the list, we're going to build a for loop

```
ip_addresses = ip_addresses.split()

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

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

This will print each entry of the list on a separate line.

```
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

Now that we have a for loop that will go through each individual item on the list, we can tweak it further to add actions to each step. By adding an if statement, we can tell it to compare the items to a parameter.

```
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)
```

Printing the updated list will see several IP's removed:

```
# Display  ip_addresses

print(ip_addresses)
```

```
['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.13
3.188', '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

Updating the file with the revised list is a matter of converting the list back to a string so that it can be written to the original file. We will first use the .join function appended to a whitespace to make the conversion.

```
# Convert `ip_addresses` back to a string so that it

ip_addresses = " ".join(ip_addresses)
```

Now that we have the variable ip_addresses defined as the list joined into a string with each entry separated by a whitespace, we will have to open the file that is to be overwritten.

We do this again using open, though this time we will use "w" for write, instead of "r" for read.

```
# Build `with` statement to read in the updated file

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

    # Read in the updated file and store the contents in `text`

    text = file.read()
```

Printing the file will now give us the updated list:

```
print(text)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219
192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

Summary

Though it is noticeably shorter than the original, I've never found this format easy to read. Before we leave let's make another for loop so we can easily compare the result to the original. We're going to use '.split' to return it to a list so we can print the entries one line at a time with a for loop.

```
text_list = text.split()

for element in text_list:
    print(element)
```

Old

```
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
```

New

```
ip_address
192.168.205.12
192.168.6.9
192.168.52.90
192.168.90.124
192.168.186.176
192.168.133.188
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.69.116
```

As you can see the IP addresses 192.168.97.225, 192.168.158.170, 192.168.201.40, and 192.168.58.57 have all been removed along with 192.168.25.60, 192.168.203.198 that were among the IP addresses added after to test the algorithm.