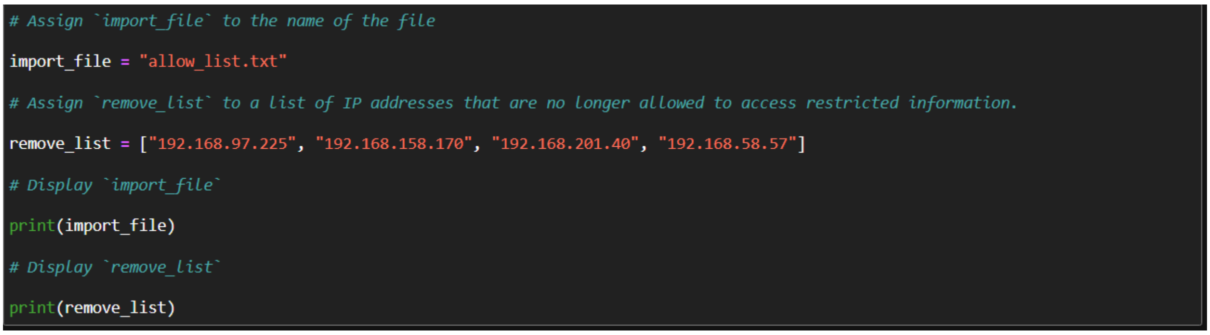
Importing and Parsing a Text File using Python

Project description

This simulation project puts the performer, Maheswar Reddy Avula, into the position of a Security Analyst for an organization. Responisibilities include developing an algorithm that parses a file containing IP addresses that are allowed to access restricted content and removes addresses that no longer than access.

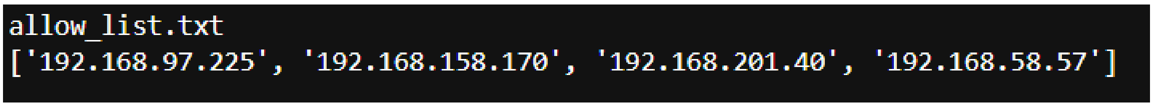
Open the file that contains the allow list

Direction: The analyst must open "allow_list.txt". Assign a string containing this file name to the import_file variable. Then, use a with statement to open it. Use the variable file to store the file while you work with it inside with statement.



```
# 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)
```

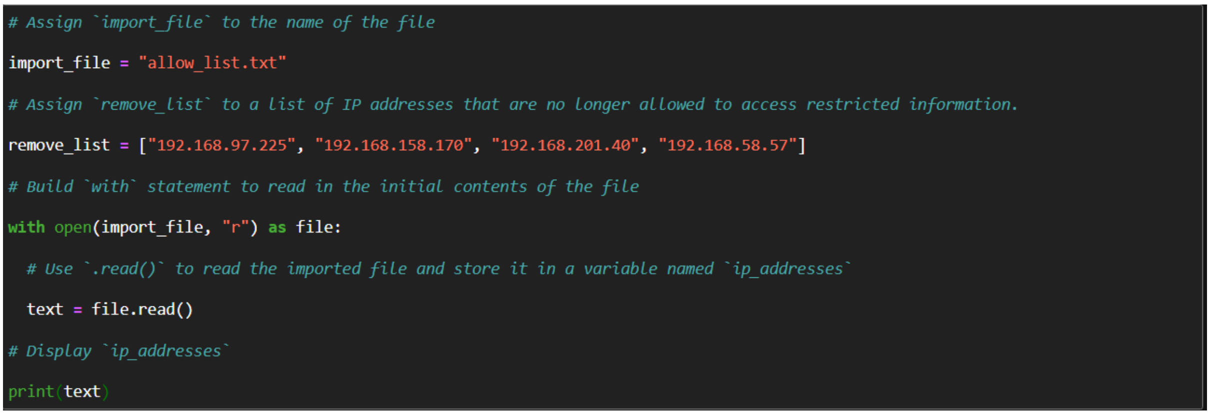
Output:



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

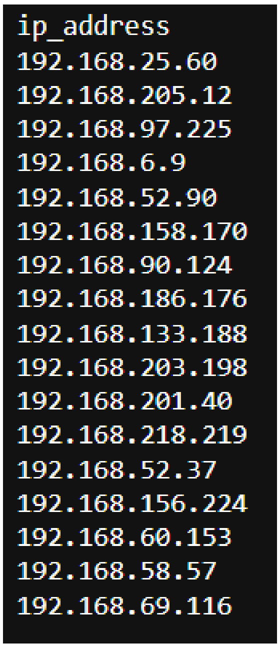
Read the file contents

Direction: The analyst must use the **.read()** method to convert the contents of the allow list file into string so that you can read them. Store this string in a variable called **ip_addresses.**



```
# 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`

  text = file.read()

# Display `ip_addresses`

print(text)
```

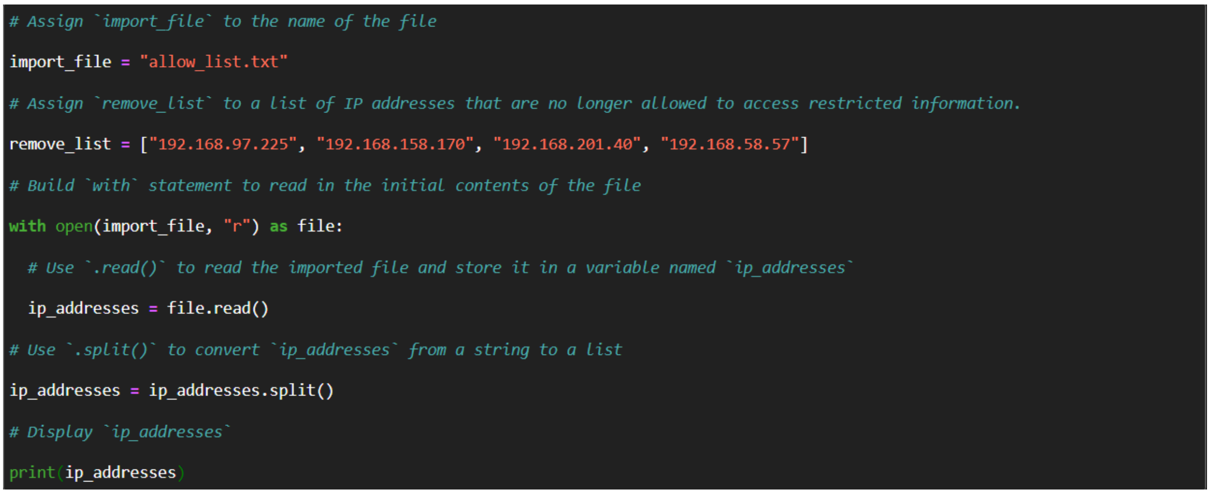
Output:



```
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
```

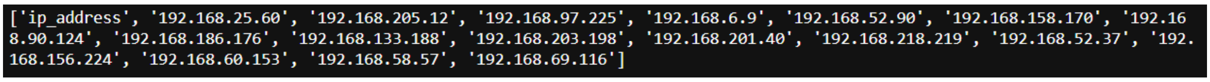
Convert the string into a list

Direction: In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, the analyst must use the **.split()** method to convert the **ip_addresses** string into a list.



```python
# 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)
```

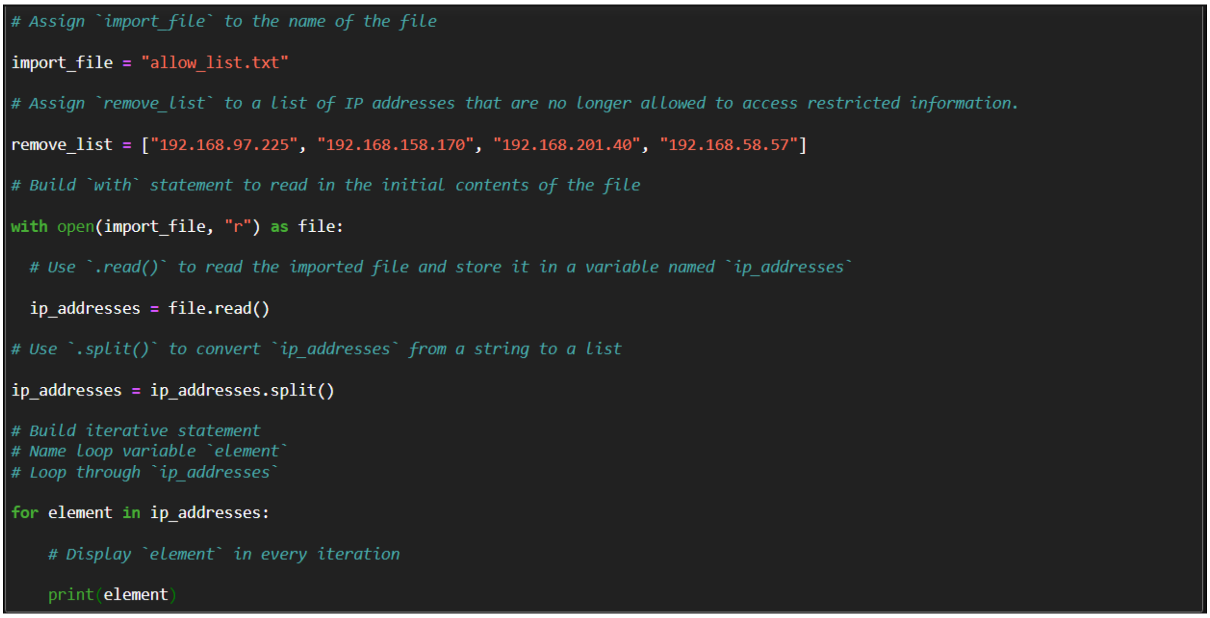
Output:



```
['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']
```

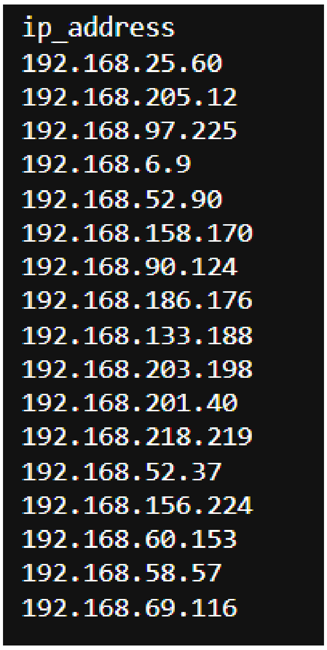
Iterate through the remove list

Direction: A second list called remove_list contains all of the IP addresses that should be removed from the ip_addresses list. The analyst must set up the header of a for loop that will iterate through the remove_list by using element as the loop variable.



```python
# 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()

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

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

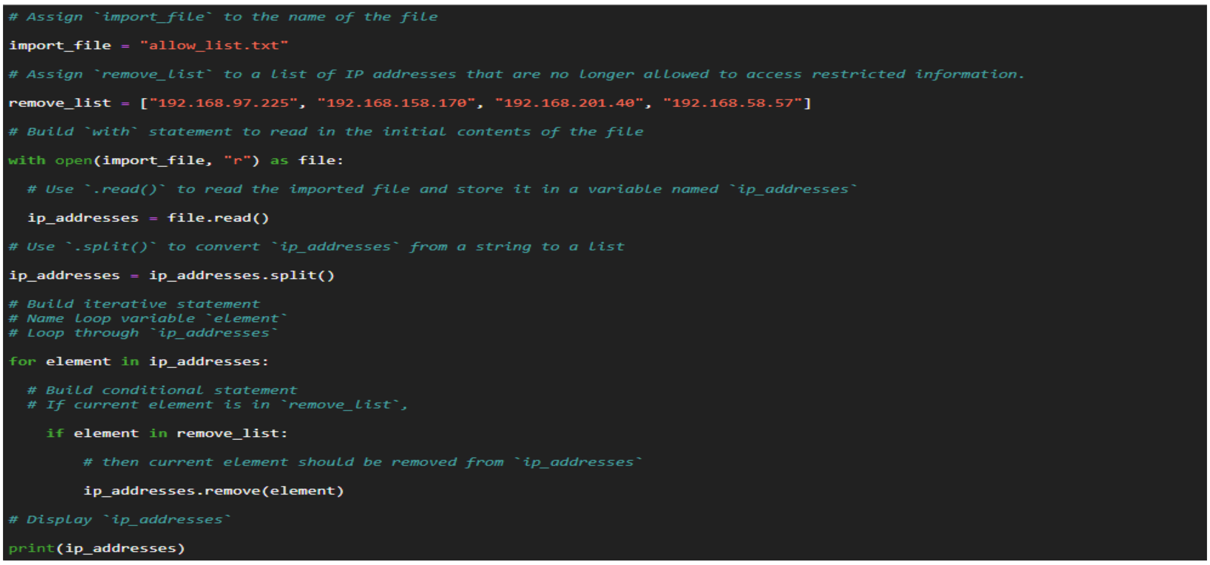
Output:



```
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
```

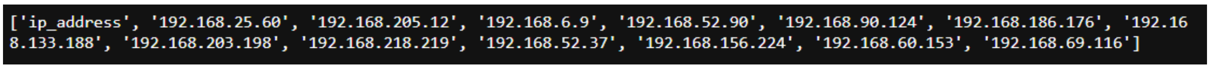
Remove IP addresses that are on the remove list

Direction: In the body of the iterative statement, the analyst must add code that will remove all the IP addresses from the allow list that are also on the remove list. First, create a conditional that evaluates if the loop variable element is part of the **ip_addresses** list. Then, within that conditional, apply the **.remove()** method to the **ip_addresses** list and remove the IP addresses identied in the loop variable element



```
# 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()

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

# Display `ip_addresses`

print(ip_addresses)
```

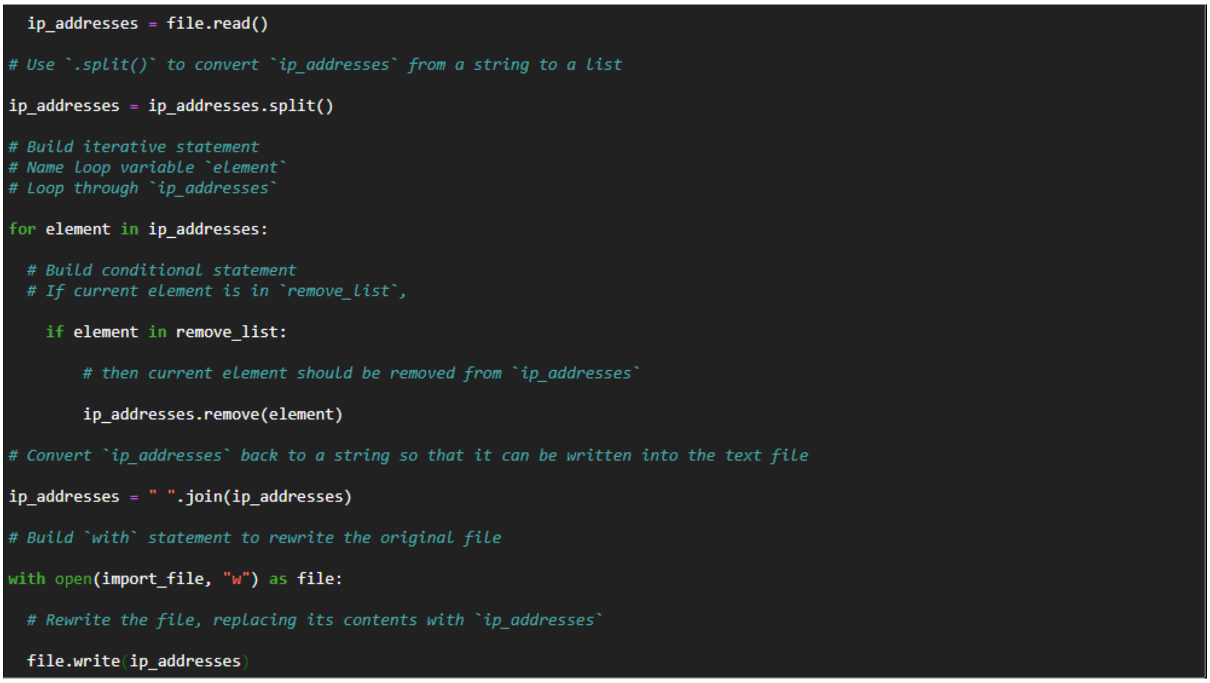
Output:



```
['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']
```

Update the file with the revised list of IP addresses

Direction: Now that these IP addresses have been removed from the **ip_address** variable, the analyst must complete the algorithm by updating the le with this revised list by rst converting the **ip_addresses** list back into a string using the **.join()** method. The, applying **.join()** to the string **"\n"** in order to separate the elements in the le by placing them on a new line. Lastly, using another with statement and the **.write()** method to write over the fiile assigned to the import_file variable.

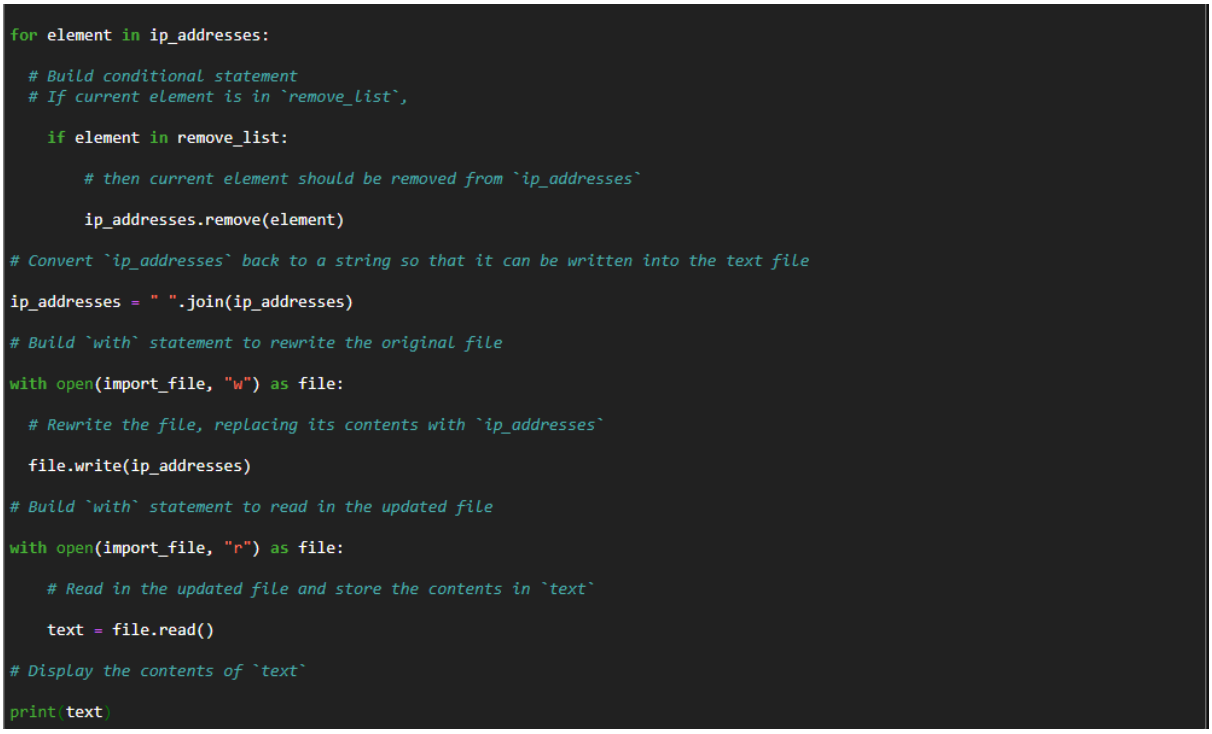


```
  ip_addresses = file.read()

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

ip_addresses = ip_addresses.split()

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

Output: NA

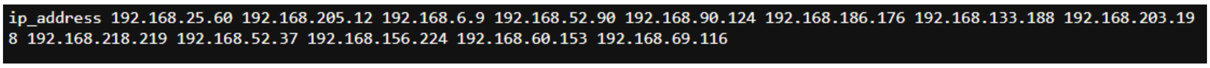
Verifying the original file was rewritten using the correct list

Direction: To verify whether the original list is rewritten with the correct list. Start by opening the file. Then read the file and store its contents in the text variable. Afterwards, display the text variable to examine the result.



```
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)

# 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()

# Display the contents of `text`

print(text)
```

Output:



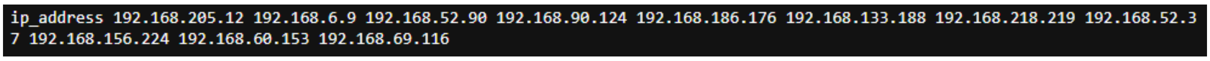
```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.203.19
8 192.168.218.219 192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

Finally required Algorithm here

```
def update_file(import_file, remove_list):

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

    ip_addresses = file.read()

  ip_addresses = ip_addresses.split()

  for element in ip_addresses:

    if element in remove_list:

      ip_addresses.remove(element)

  ip_addresses = " ".join(ip_addresses)

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

    file.write(ip_addresses)
```

```
update_file("allow_list.txt",["192.168.25.60","192.168.140.81""192.168.203.198"])
with open("allow_list.txt", "r") as file:
  text = file.read()
print(text)
```

Output:



```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.3
7 192.168.156.224 192.168.60.153 192.168.69.116
```

Summary

The required algorithm that parses a le containing IP addresses that are allowed to access restricted content and removes addresses that no longer have access has been successfully made. All tasks were successfully completed in accordance with the directions given by the organization.