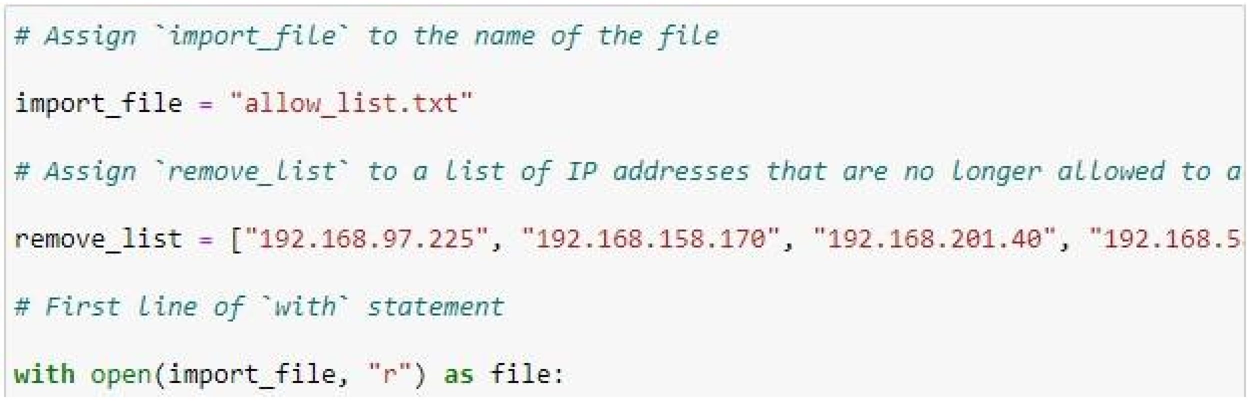
Algorithm for file updates in Python

Project description

I am a security professional working at a health care company. I'm required to regularly update a file that identifies the employees who can access restricted content. Employees are restricted access based on their IP address. There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from the allow list. I am to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, I will remove those IP addresses from the file containing the allow list.

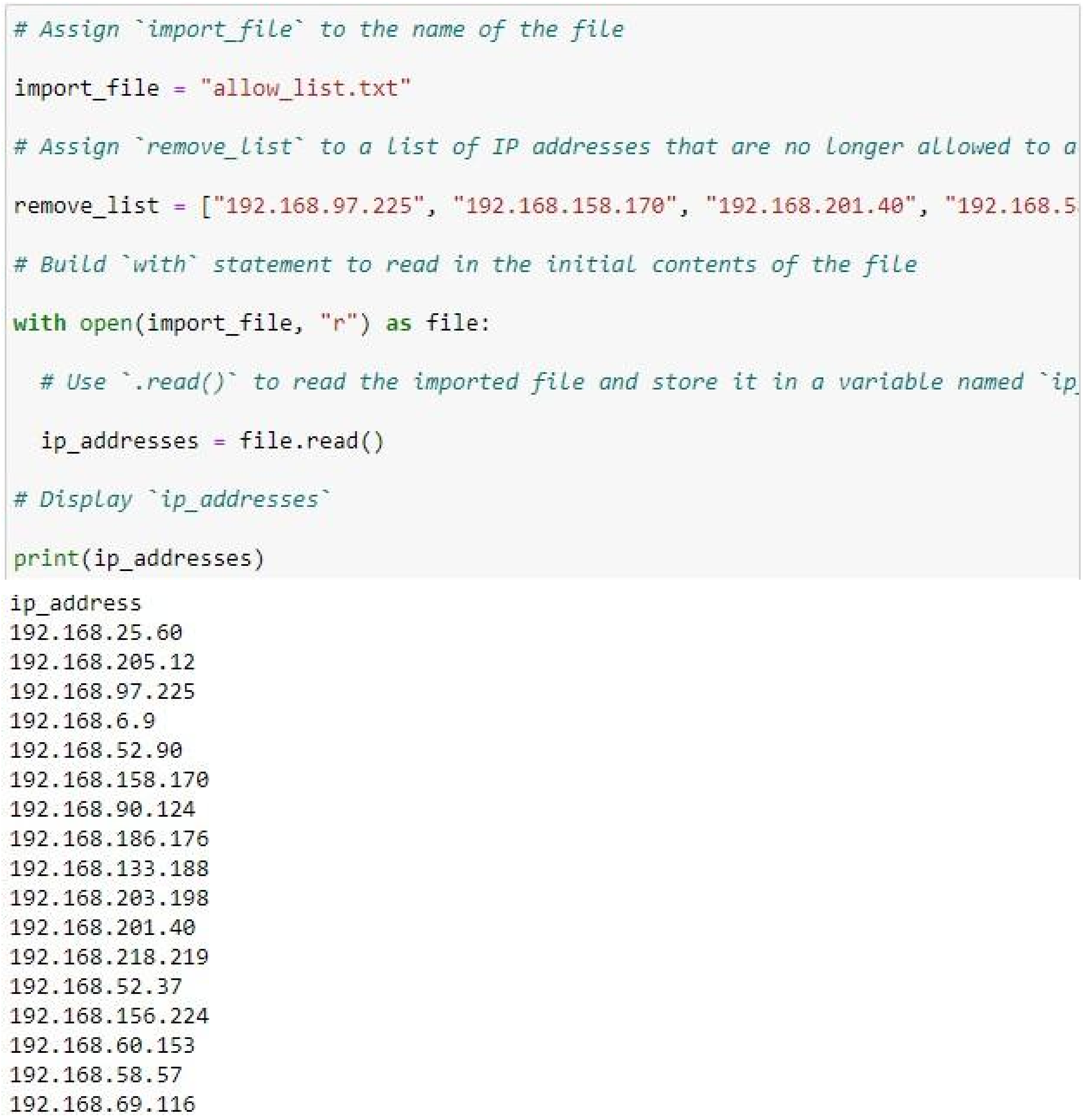
Open the file that contains the allow list



```
# 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 a

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.5

# First line of `with` statement

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

I used a with statement to edit this file. In the with statement, I used the open() method to open the file and close once I am done editing the file. Inside of the open() method, I have two parameters. The first parameter is the name of the file in which I am editing and the second parameter is what I want to do to the file, which is "r" for read. I then use the word "as" to let Python know to save the newly created file with the name "file". I then ended the statement with the colon(:).

Read the file contents



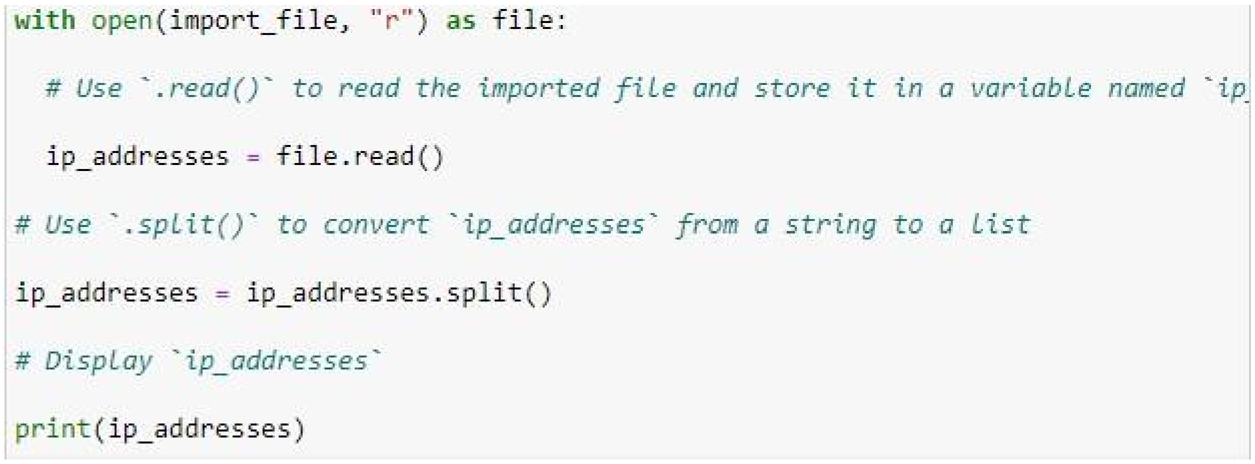
```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to a

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.5

# 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_

  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
```

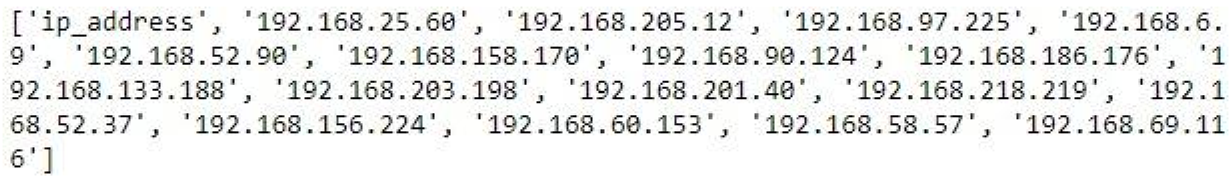
In addition to the previous task, I used the .read() method to read the contents of the newly created file then printed the contents of the file to the screen.

Convert the string into a list

[Add content here.]



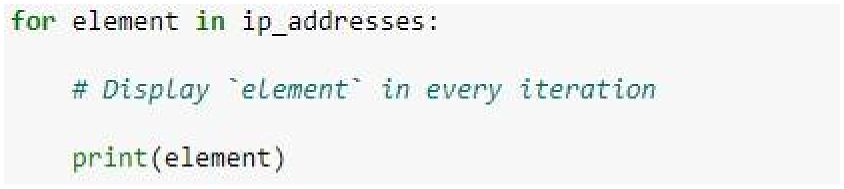
```
with open(import_file, "r") as file:

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

  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)
```



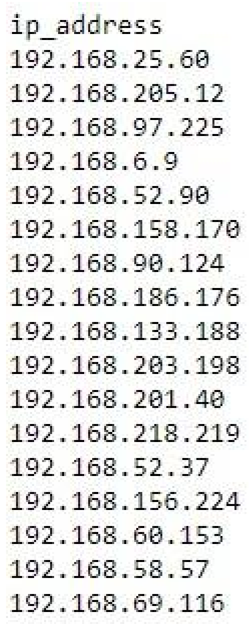
```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.
9', '192.168.52.90', '192.168.158.170', '192.168.90.124', '192.168.186.176', '1
92.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.1
68.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.11
6']
```

In this task, I used the .split() method to convert the string to a list.

Iterate through the remove list



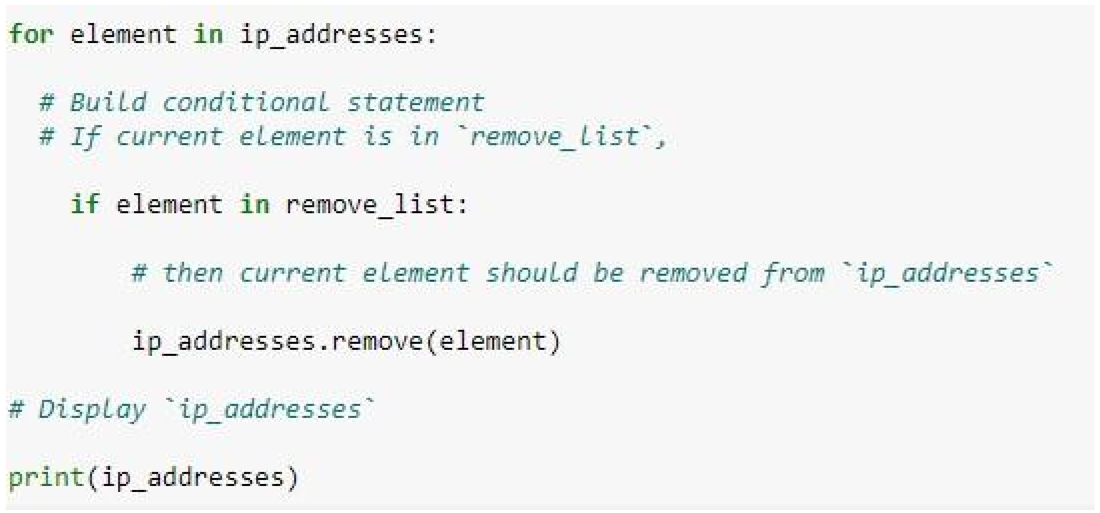
```
for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```



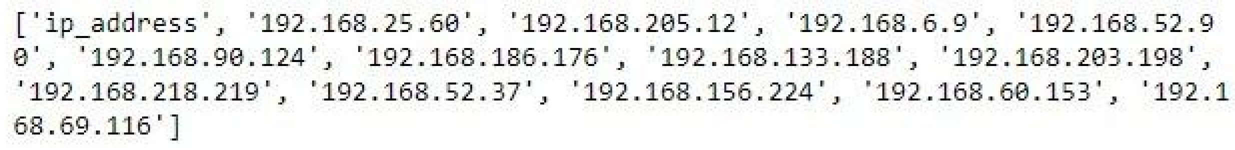
```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

I created this for loop to iterate through the remove list. The keyword “for” initiates the for loop. I used the keyword “element” as the loop variable and used “in” as the loop condition.

Remove IP addresses that are on the remove list



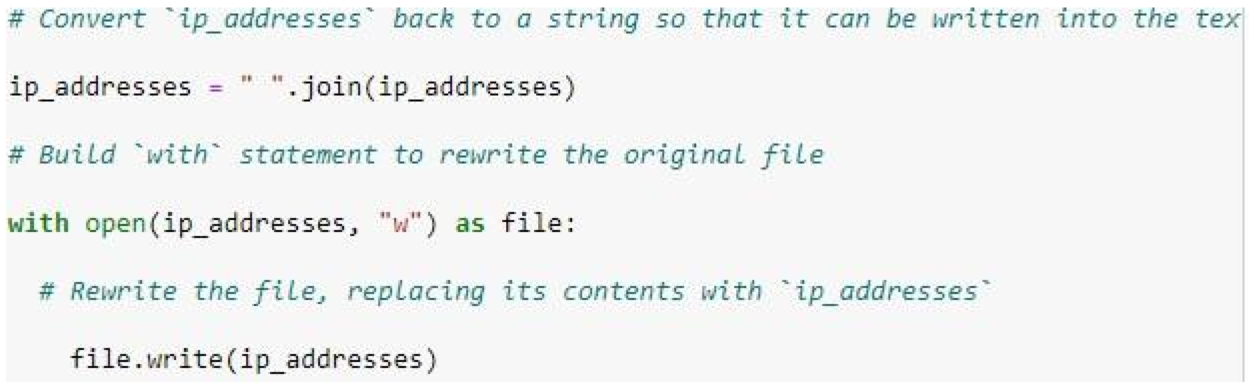
```
for element in ip_addresses:

  # 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', '192.168.52.9
0', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198',
'192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.1
68.69.116']
```

To remove the IP addresses that are on the remove list, I used the .remove() method.

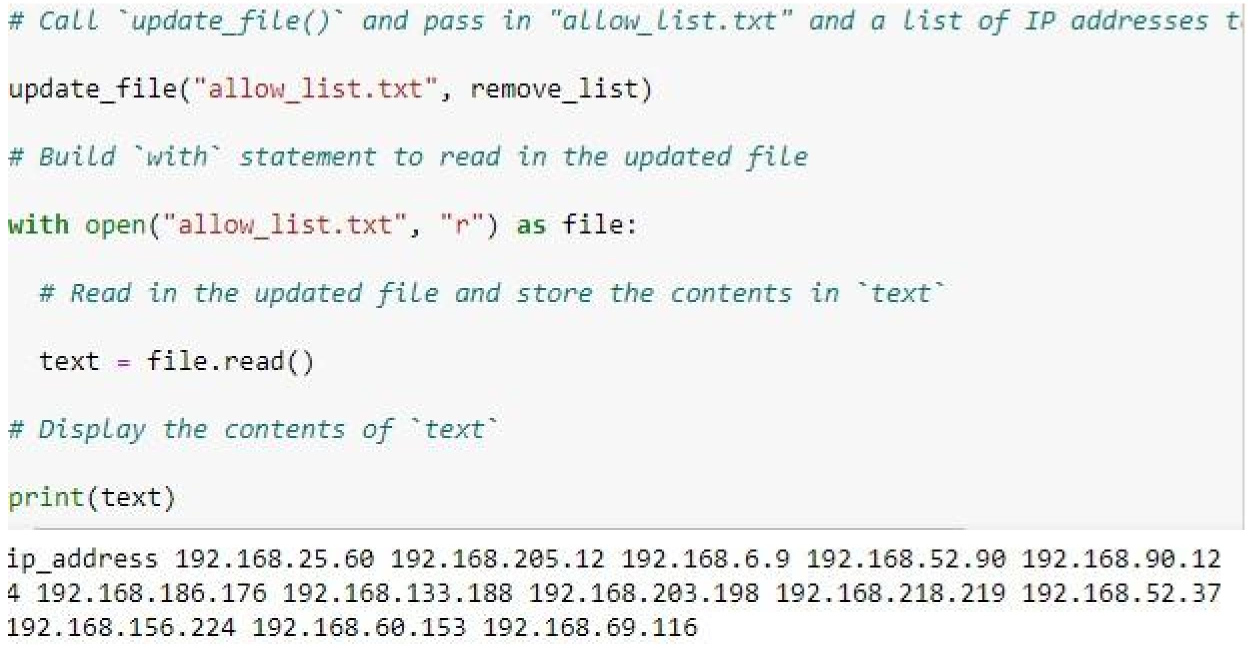
Update the file with the revised list of IP addresses



```
# Convert `ip_addresses` back to a string so that it can be written into the tex

ip_addresses = " ".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)
```

In this task, I used another with statement. In this with statement, however, I used the “w” as a second parameter in the open() method. This is specifying that I’m opening the file for the purposes of writing to it.

Summary



```
# Call `update_file()` and pass in "allow_list.txt" and a list of IP addresses t

update_file("allow_list.txt", remove_list)

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

with open("allow_list.txt", "r") as file:

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

  text = file.read()

# Display the contents of `text`

print(text)
```

```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.12
4 192.168.186.176 192.168.133.188 192.168.203.198 192.168.218.219 192.168.52.37
192.168.156.224 192.168.60.153 192.168.69.116
```

This shows an update of the new file excluding the IP Addresses that are on the remove_list file.