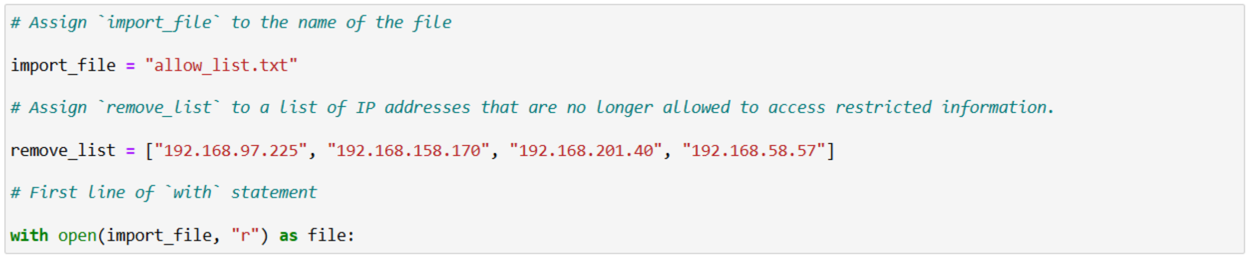
Algorithm for file updates in Python

Project description

The goal is to develop an algorithm that parses a series of IP addresses that have access to restricted information and removes the addresses that are no longer allowed. Python can automate this process.
The project assumes a text file called `"allow_list.txt"` that contains a series of IP addresses that are allowed to access restricted information. There are IP addresses that should no longer have access to this information, and their IP addresses need to be removed from the text file. The instructions provided a variable named `remove_list` that contains the list of IP addresses to be removed.

Open the file that contains the allow list

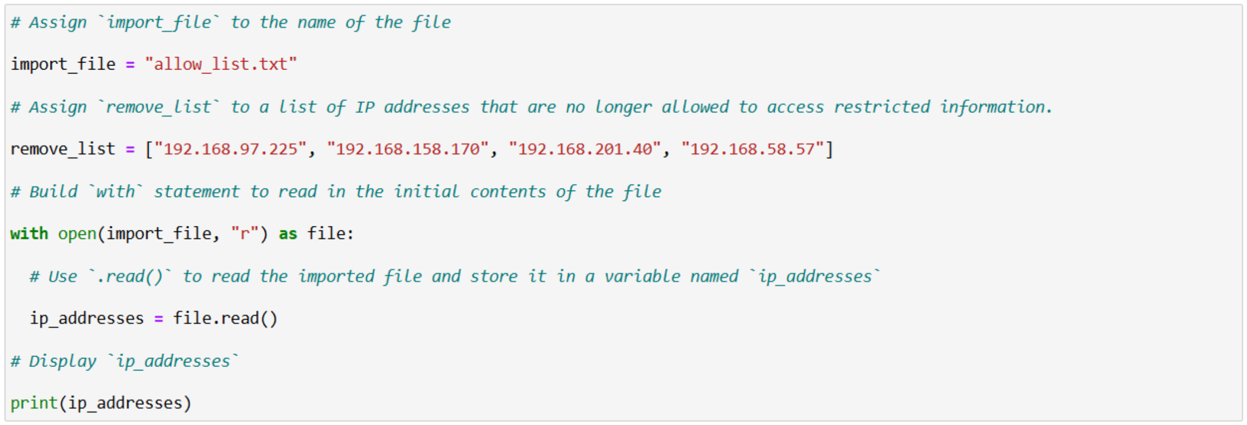
The file that you want to open is called `"allow_list.txt"`. Assign a string containing this file name to the `import_file` variable. Then, use a `with` statement to open it. Use the variable `file` to store the file while you work with it inside the with statement.



```
# 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"]

# First line of `with` statement

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

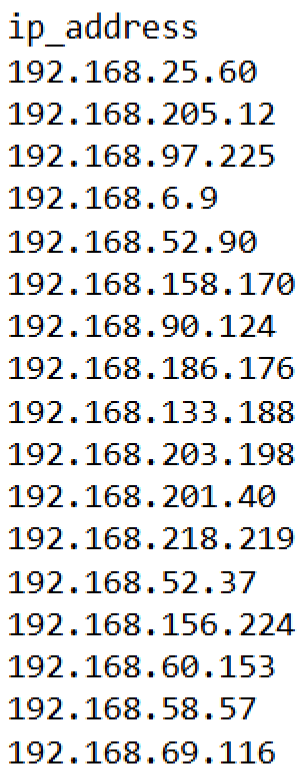
Read the file contents

Next, use the `.read()` method to convert the contents of the allow list file into a 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`

  ip_addresses = file.read()

# 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
192.168.90.124
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

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

A second list called `remove_list` contains all of the IP addresses that should be removed from the `ip_addresses` list. Set up the header of a for loop that will iterate through the `remove_list`. Use `element` as the loop variable.

```
# 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

In the body of your iterative statement, 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 identified in the loop variable element.

```
# 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

Now that you have removed these IP addresses from the `ip_address` variable, you can complete the algorithm by updating the file with this revised list. To do this, you must first convert the `ip_addresses` list back into a string using the `.join()` method. Apply `.join()` to the string `"\n"` in order to separate the elements in the file by placing them on a new line.
Then, use another `with` statement and the `.write()` method to write over the file assigned to the `import_file` variable.

```python
# 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)
```

Summary

The instructions for the original project walked us through a specific step-by-step build, but not the most efficient way to complete the project goals. Instead of iterating through all the addresses on the allowed IPs list, it would be far more efficient to iterate through the remove list. In addition, this should actually be separated into an independent function so that it can be called repeatedly, should there be new addresses that need to be removed.
The revised code should look thus:

```
# Create a function called “ip_remove” with expected parameters “import_file” and
“remove_list”

def ip_remove(import_file, remove_list):

    # Open the import file, save it as local variable “ip_addresses”

    with open(import_file, “r”) as file:
        ip_addresses = file.read()

    # Convert the ip_adresses string into a list type

    ip_addresses = ip_addresses.split()
    # Iterate through the list of addresses to be removed, checking if it’s in the allow
list.
    # If found on the allow list, remove it.

    for i in remove_list:

        if i in ip_addresses:

            ip_addresses.remove(i)

    # Convert ip_addresses back from list to a string, one address per line:

    ip_addresses = “\n”.join(ip_addresses)

    # Write the string back to the original import file

    with open(import_file, “w”) as file:

        file.write(ip_addresses)

# Now that the function has been defined, it can be run with a few selected parameters:
# We will import the file “allow_list.txt”
# The IP addresses to be removed are stored in the variable ips_to_remove

ips_to_remove = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

ip_remove(“allow.txt”, ips_to_remove)
```