Algorithm for file updates in Python

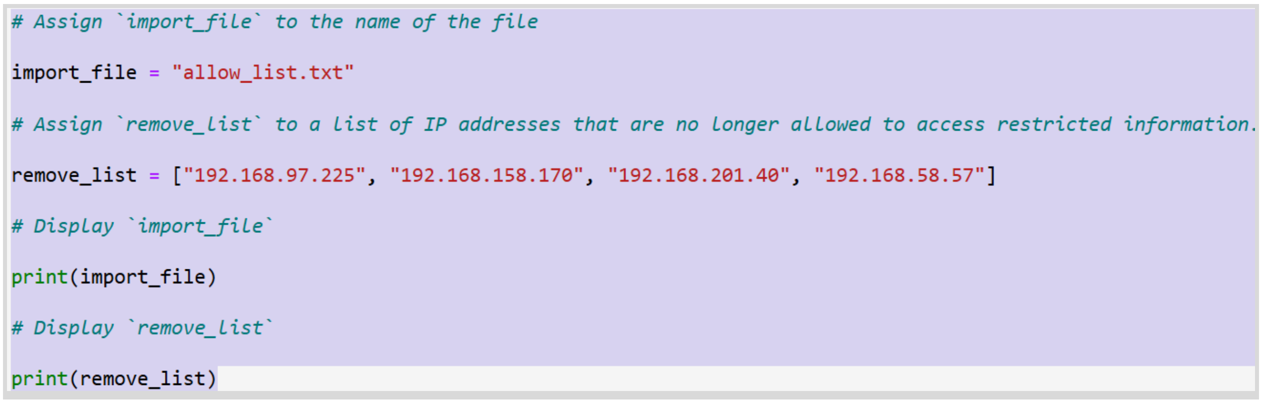
Project description

As a security professional working at a health care company I was required to regularly update a file that identified the employees who could access restricted content. The contents of the file were based on who was working with personal patient records. Employees were restricted access based on their IP address. There was an allow list for IP addresses permitted to sign into the restricted subnetwork. There was also a remove list that identified which employees I must remove from this allow list. My task was to create an algorithm that used Python code to check whether the allow list contained any IP addresses identified on the remove list.

The IPs removed:

- `192.168.97.225`
- `192.168.158.170`
- `192.168.201.40`
- `192.168.58.57`

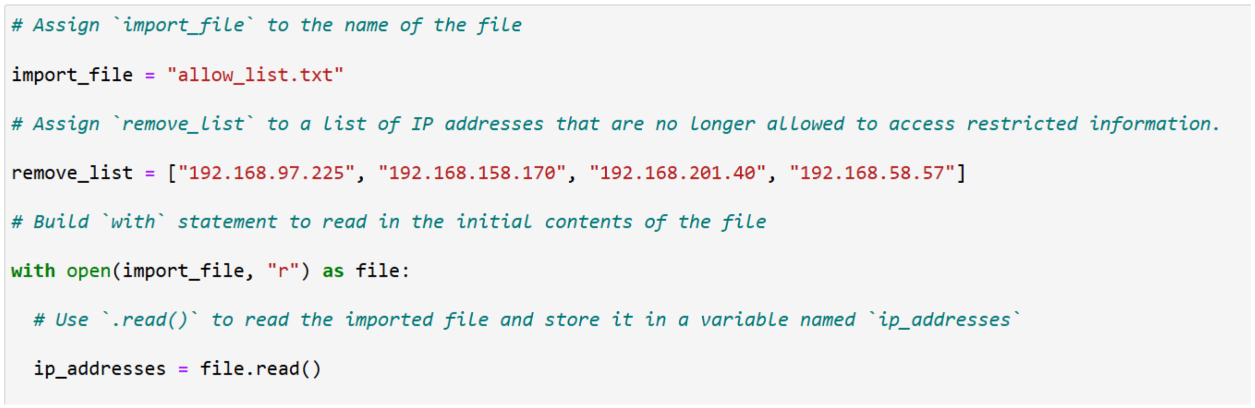
Open the file that contains the allow list



```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"]

# Display `import_file`

print(import_file)

# Display `remove_list`

print(remove_list)
```

Read the file contents

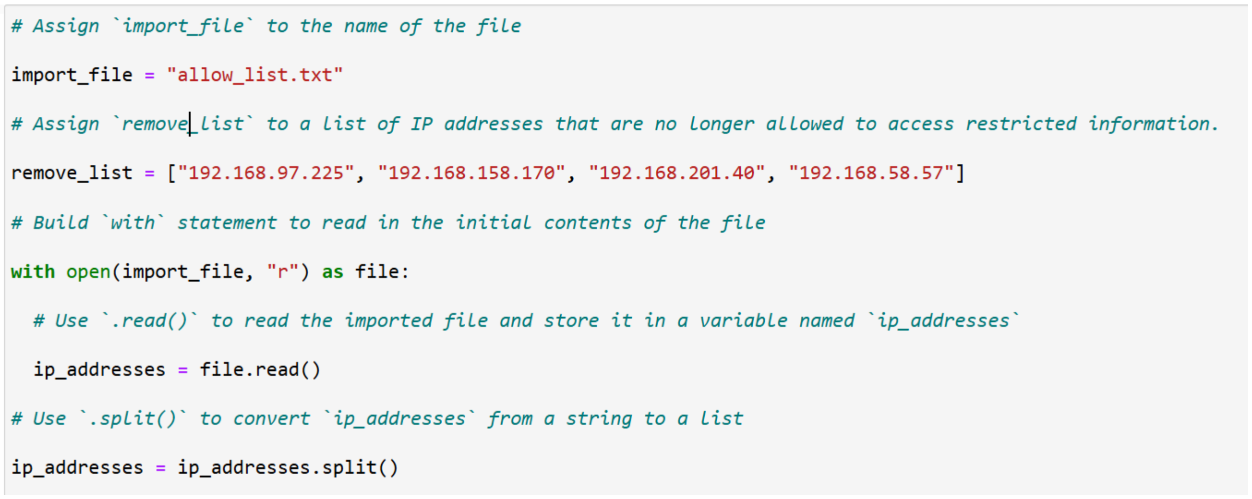
I used the `.read()` method to read the imported file and stored its contents in a variable named `ip_addresses`.



```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()
```

Convert the string into a list

After reading the file I reassigned the `ip_addresses` variable to a list data type using the `.split()` method. This allowed me to iterate through each individual IP address in the allow list, instead of processing a large string containing all addresses.



```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()
```

Iterate through the remove list

First, I built the iterative statement. Named the loop variable element, looped through ip_addresses, and displayed each element.

```
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)
```

Remove IP addresses that are on the remove list

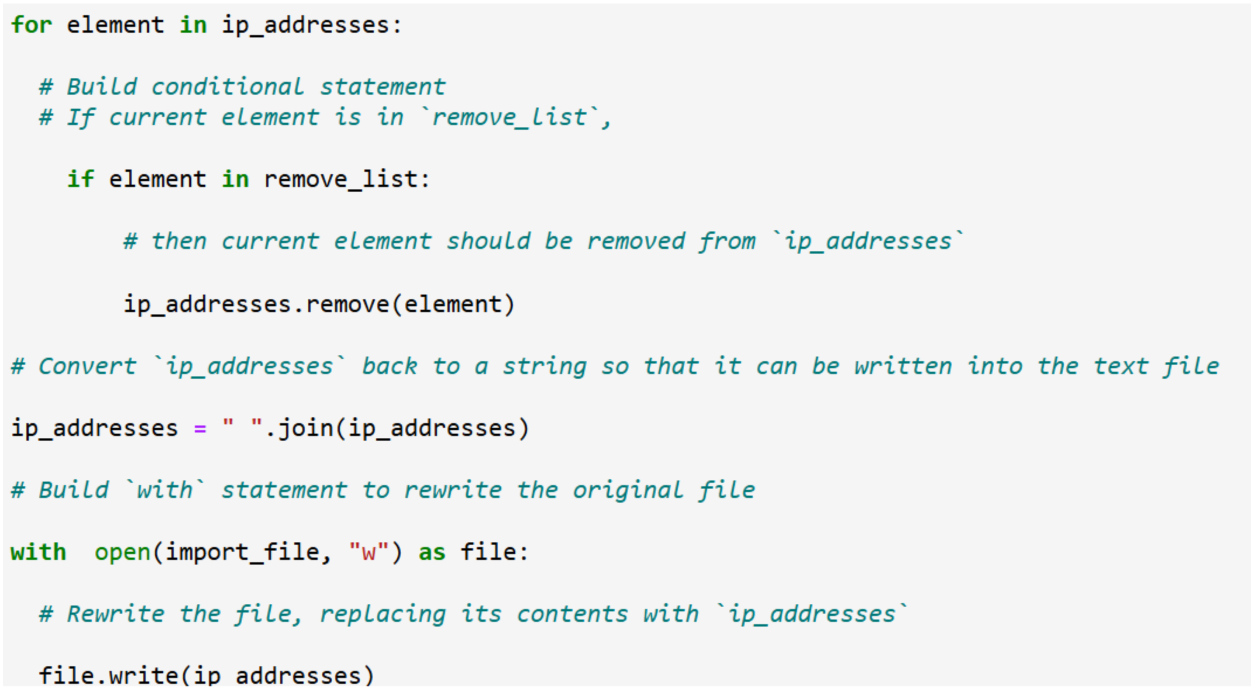
Now, I built a conditional statement to remove the elements of `remove_list` from the `ip_addresses` list. The conditional statement was placed inside the iterative statement that looped through `ip_addresses`. In every iteration, if the current element in the `ip_addresses` list was in the `remove_list`, the `remove()` method was used to remove that 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)
```

Update the file with the revised list of IP addresses

The next step was to update the original file that was used to create the `ip_addresses` list. A line of code containing the `.join()` method was added to the code so that the file could be updated. This was necessary because `ip_addresses` had to be in string format when used inside the `with` statement to rewrite the file.

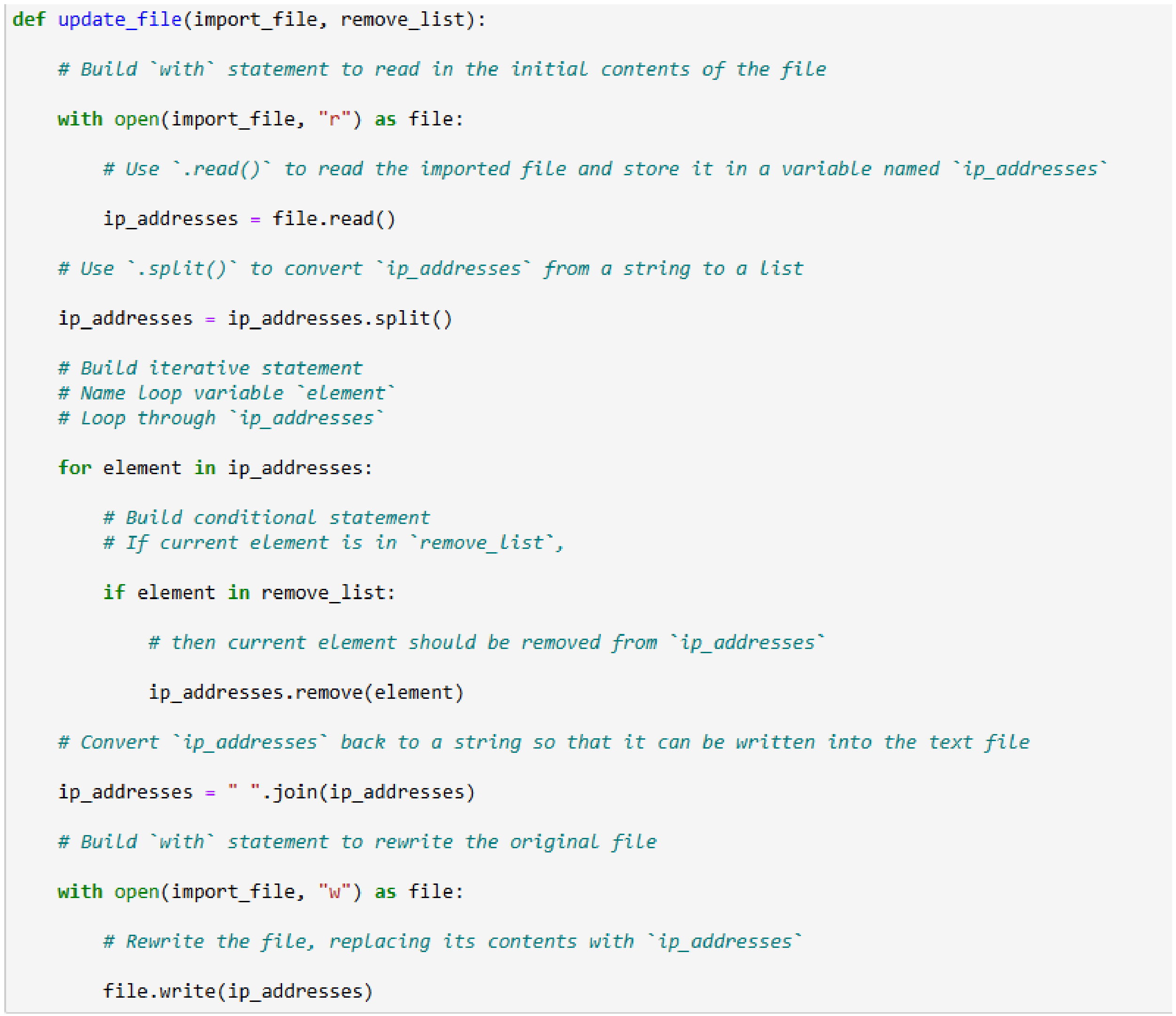
Next I built the with statement that rewrote the original file. I used the `"w"` parameter when calling the `open()` function to delete the contents in the original file and replace them with what I wanted to write.



```
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)
```

Create a function to automate the process

Finally I defined a function named `update_file()` that took in two parameters. The first parameter was the name of the text file that contained IP addresses (named `import_file`). The second parameter was a list that contained IP addresses to be removed (named `remove_list`).



```python
def update_file(import_file, remove_list):

    # 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)

    # 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

First I imported the `"allow_list.txt"` file into the Python environment. Then, I opened the file containing the allow list, read its contents, and converted the string into a list. Next, I iterated through the remove list and removed any IP addresses present in both lists. Finally, I updated the file with the revised list of IP addresses and created a function to automate the process by implementing it wherever necessary.