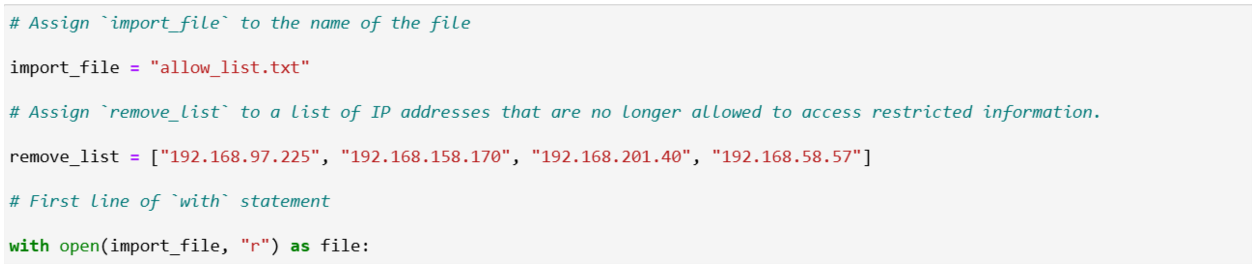
# Algorithm for file updates in Python

## Project description

I am a security analyst working at a healthcare company. Part of my job is regularly updating a file identifying employees who can access restricted content. The file’s contents are based on who works with personal patient records. Employees are restricted access based on their IP address. There is a list of IP addresses permitted to sign into the restricted subnetwork. There is also a list that identifies which employees must be removed from this allowed list. My task was 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, it should remove those IP addresses from the allowed list file.

## Open the file that contains the allow list

First, I must read the `“allow_list.txt”` file to get the current list of allowed IP addresses. The text file is stored in the `import_file` variable so it can be reused later on in the program. Using the keyword `with` and `open()` function to open the file and with `import_file` as the first parameter and second parameter `“r”` to allow reading privileges. Afterward, the keywords `as file:` assign `file` variable as the reference for the `open()` function output.



```
# 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

The next line should be indented because we want to read the now open text file. Then use the `.read()` method on the `file` to convert the allowed list into strings and store it in the variable `ip_addresses` to update it later in the program. To check that the list is stored we go to the next line and use code `print(ip_addresses)` and see it prints out a string list of 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)
```

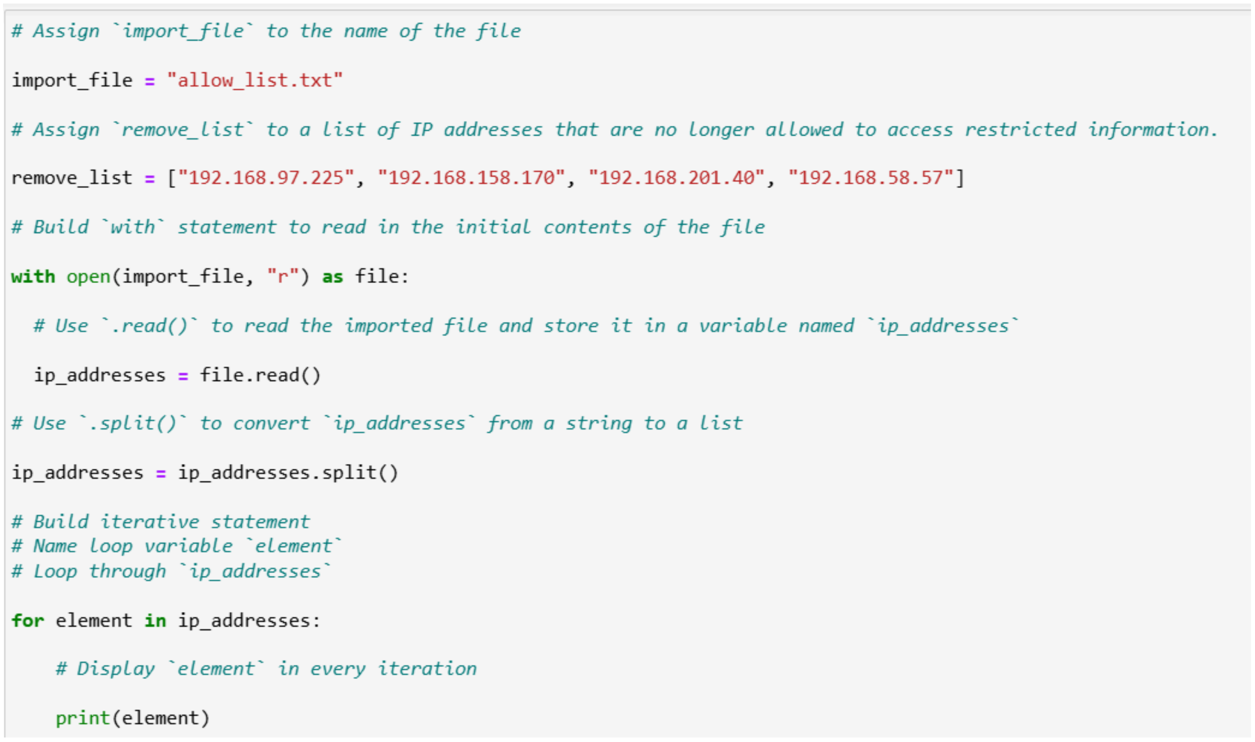
## Convert the string into a list

It is easier to read the IP addresses as a list rather than one large string because of the separation of each element. I used the `.split()` method which converts the string into a list with whitespace as the breaking points and stored it back in `ip_addresses`. Used code `print(ip_addresses)` to double-check that the string was now 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)
```

## Iterate through the remove list

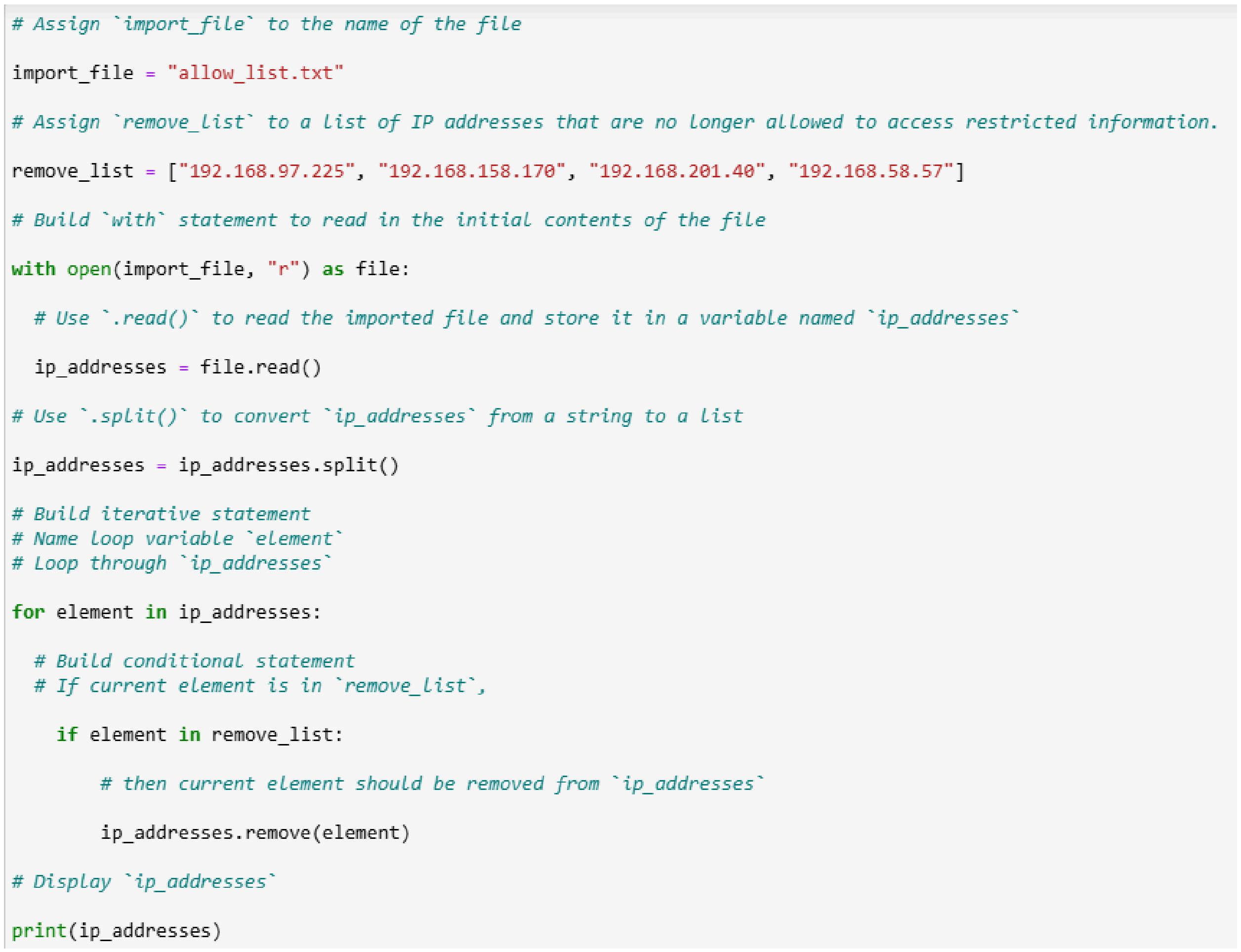
Before removing the `remove_list` IP addresses I need to build the loop that iterates through the `ip_addresses` list. Using a `for` loop with `element` variable it will iterate through each IP address in the `ip_addresses` making it easier to find the specific ones.



```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)
```

## Remove IP addresses that are on the remove list

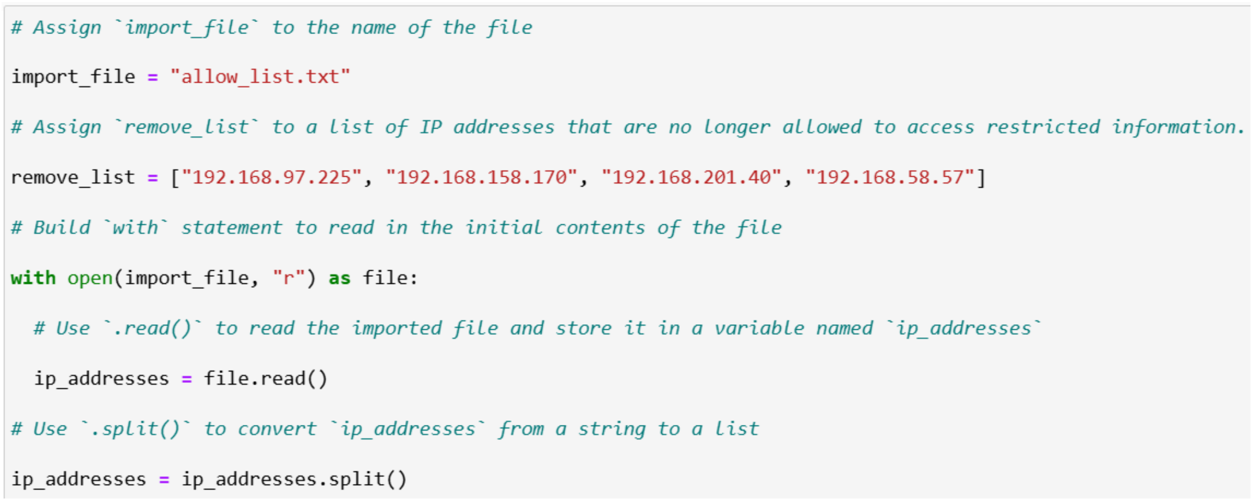
In order to see if the current `element` in the loop is in the `remove_list` I must use a conditional statement. Using an if statement I can check that if the `element` is in `remove_list` use the `.remove()` method to remove it from the list. Used code `print(ip_addresses)` to double-check that the necessary IP addresses were removed from the `ip_addresses`.



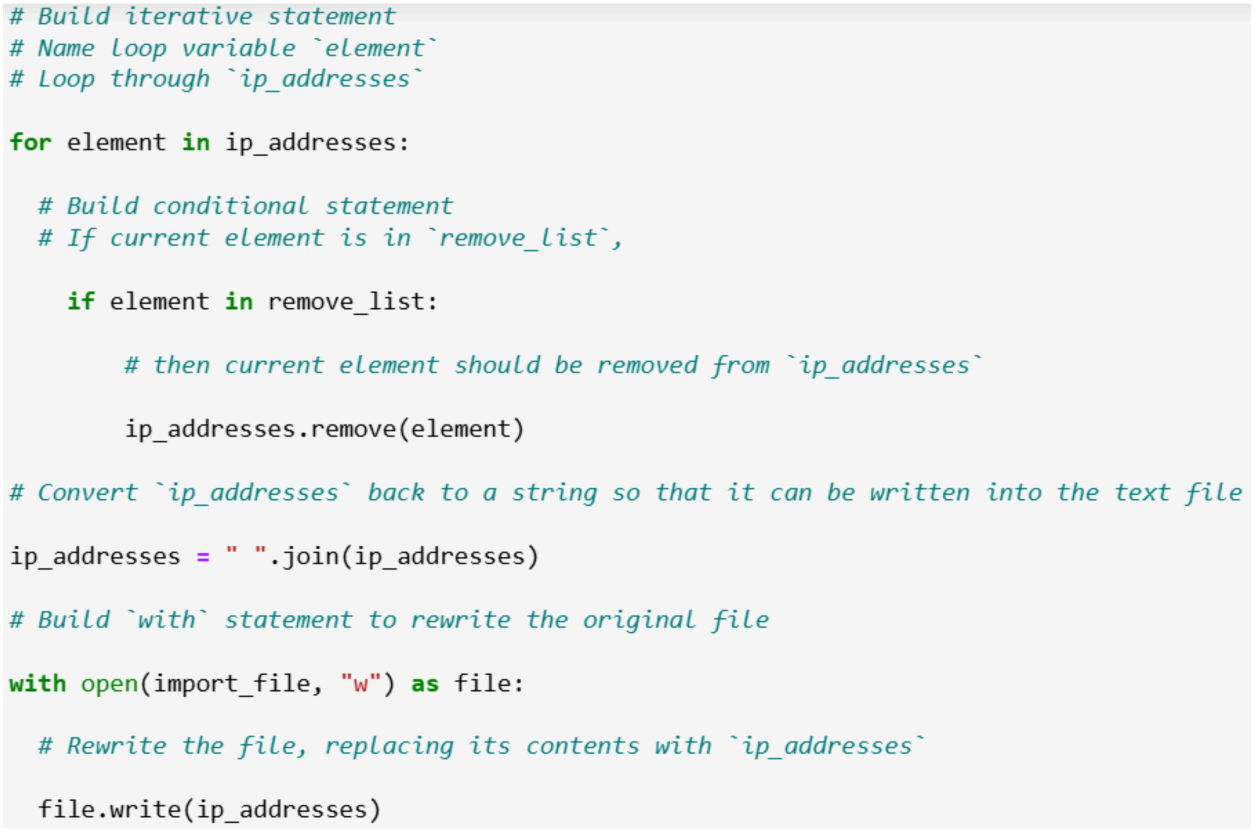
```python
# Assign `import_file` to the name of the file

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()

# 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)
```

## Update the file with the revised list of IP addresses

To update the text file with this new list I need to convert it back to a string and write it to the file. Using the `.join()` method with the syntax `“ “` and `ip_addresses` I converted the list into a string with white space separating the IP addresses. Next, I reopened the file with the code `with open(import_file, “w”) with file:` with `“w”` allowing writing privileges to the file. For the next line, I indented and used the `.write()` method with `ip_addresses` to update the file with the new 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()
```



```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

As a security analyst at a healthcare company, I was tasked with updating the list of IP addresses allowed to access restricted personal patient files. I was able to use the codes `with open(import_file, “r”) as file:` and `file.read()` to store the file output as a string in the variable `ip_addresses`. Next, I used the `.split()` method to convert the string in `ip_addresses` into a list to find the necessary IP addresses more easily. I created an algorithm using a `for` loop with an `element` variable to iterate through the `ip_addresses` list and an `if` statement that checks if the current IP address is in the `remove_list`. If the `element` were in the list I used the `.remove()` method to remove it, else it would move on. After the loop finished, I converted the list back to a string with the `.join()` method and used the codes `with open(import_file, “w”) as file:` and `file.write(ip_addresses)` to update the file to the new allowed IP addresses list.