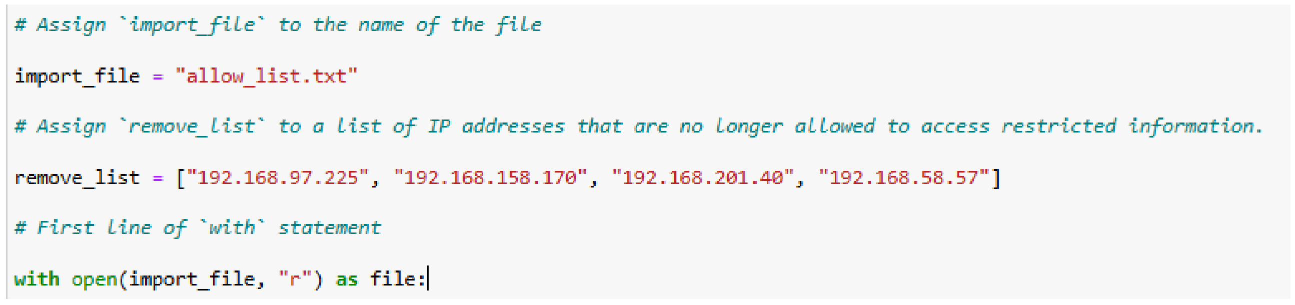
Algorithm for file updates in Python

Project description

This task demonstrates what method a professional working at a healthcare company might choose to update files for security purposes. Fundamental to this role is identifying which employees can access patient records. This is managed through IP addresses - these are featured on allow or remove lists which identify employees that have access and should no longer have access, respectively. Here, an algorithm is implemented to check whether the allow list contains any IP addresses contained in the remove list and if so, removes these from the file containing the allow list.

Open the file that contains the allow list

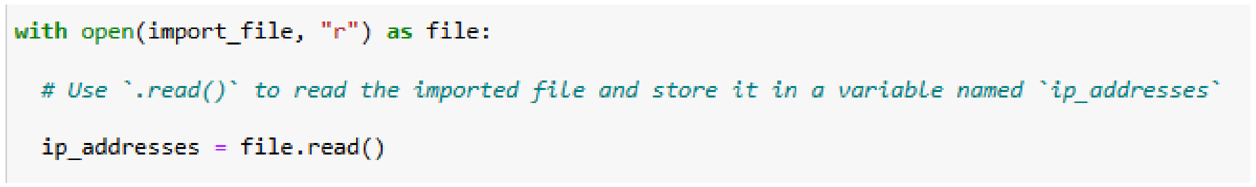
In order to open this .txt file, 'with' needs to be specified before the 'open' function. Python automatically closes the file once the code block inside 'with' is completed. This is vital for file management, otherwise problems such as memory leaks can occur if the file is not closed. The file, in this case, allow_list.txt, was assigned to a variable before specifying the read parameter, "r" which means 'read'.



```
# 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

As mentioned, the 'r' parameter included after the file name indicates the user wishes to read the file. Following the opening of the file, a variable "ip_addresses" is used to read and store the imported 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

We can use the .split() method to convert the IP addresses from a string to a list. By default, when nothing is included within the .split() parenthesis, each element within a list is separated when there is the presence of whitespace in the .txt file.

```
# 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', '192.16
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

This for loop identifies each ip address within ip_addresses through iteration and prints each one.

```
for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

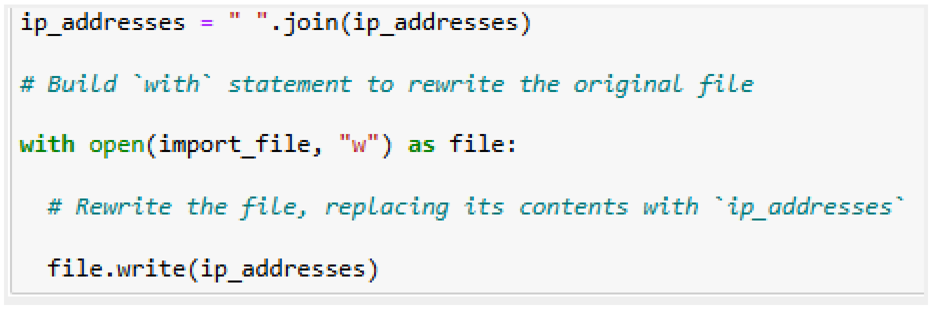
Remove IP addresses that are on the remove list

Here, the for loop implements a condition evaluating if the loop variable "element" is part of the ip_addresses list. Inside that conditional, the application of the .remove() method to the ip_addresses list removes those elements from ip_addresses that were also identified within 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)
```

Update the file with the revised list of IP addresses

With the use of .join(), the user can return the IP address information from a list back to string format so that it can be written to a text file. Now the ip_addresses are back to string format they are written to the import_file (“w” parameter and “file.write”)



```
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

To summarise, this implementation of an algorithm in python has employed some useful methods to update information. These are:

- read() and write(), in order to open, read the current information and overwrite it.
- .split() to convert string data from the .txt file to a list. This means the lists could be iterated over with a for loop, identifying all separate IP addresses and removing the ones that were no longer allowed using .remove().
- .join() was used to complete the algorithm - the updated IP addresses were converted back from a list to a string so that it could overwrite the original file.