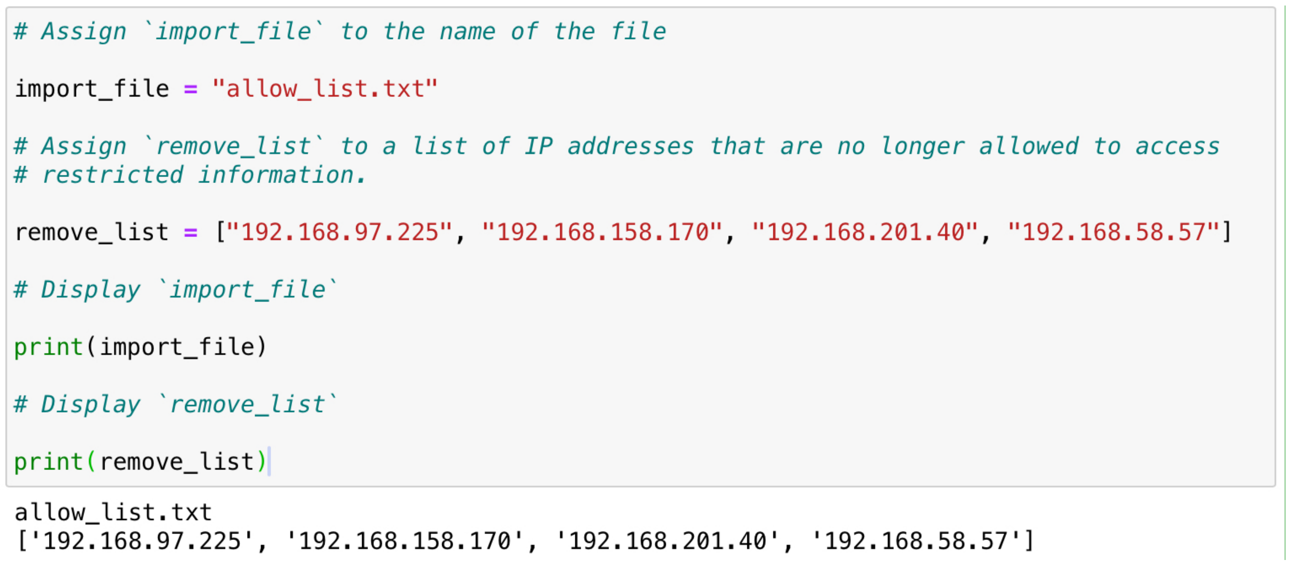
Algorithm for File Updates in Python

Project Description

I take on the role of a security professional managing access to restricted patient records in a healthcare company. Access is managed via an access control list that lists the IP addresses of users allowed access to patient records. In this task I update this list by removing IP addresses listed in a separate removal list in order to maintain security of patient records. I use Python to automate this process.

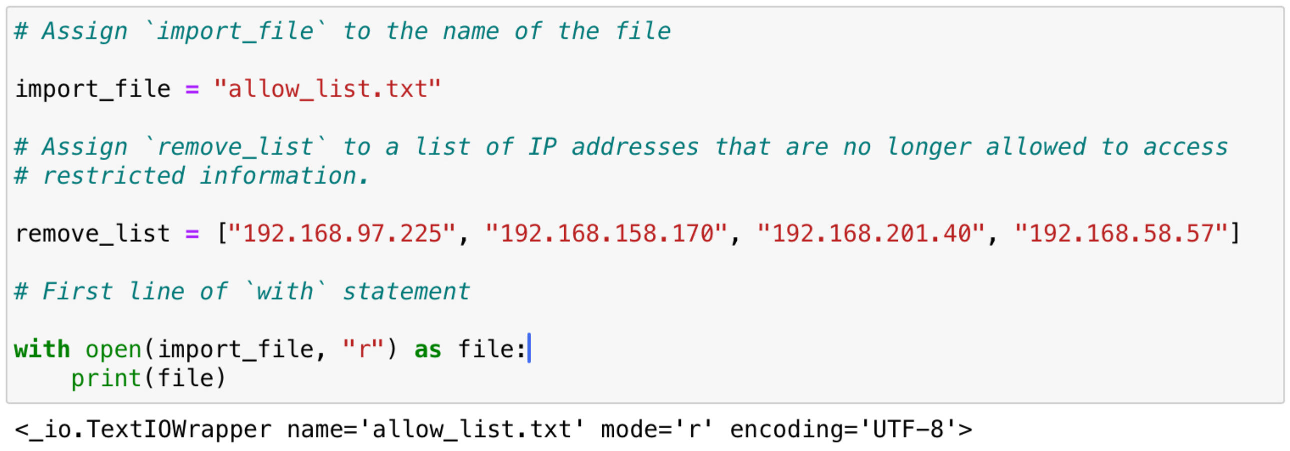
Open the file that contains the allow list



```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)
```

```
allow_list.txt
['192.168.97.225', '192.168.158.170', '192.168.201.40', '192.168.58.57']
```

First I print the contents of the variables to check I have entered them correctly and see how Python interprets them.



```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"]

# First line of `with` statement

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

```
<_io.TextIOWrapper name='allow_list.txt' mode='r' encoding='UTF-8'>
```

The with command is used to open the allow_list.txt document as a file. Python treats the allow_list.txt as a UTF-8 encoded file in read mode, because the .open() command specified the “r” parameter. This parameter can also be “a” if appending (adding text to the end of existing file contents without erasing the existing contents), or “w” if overwriting.

Read the file contents

```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:
  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
```

The .read() command reads the contents of the file into the variable ip_addresses. I then display the variable ip_addresses to check the contents of the file.

The file allow_list.txt contains a list of IP addresses. Notice the 4 IP addresses in the remove_list are all present at first.

Convert the 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)
```

```
['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', '192.168.156.22
4', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

I convert the IP addresses in the allow_list.txt file into a Python list using the .split() command. This provides me with a list I can later iterate through to check for elements to be removed, and allows me to use the .remove() command later to remove such elements from the list.

Iterate through elements in the allow list

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

import_file = "allow_list.txt"
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)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
```

I next practice iterating through elements in the allow_list.txt file, which has been loaded into the ip_addresses variable.

The for command iterates through elements in the ip_addresses list using a for loop. In each iteration the element is printed out.

A truncated list of the first 3 IP addresses is shown in the screenshot.

Remove IP addresses that are on the remove list

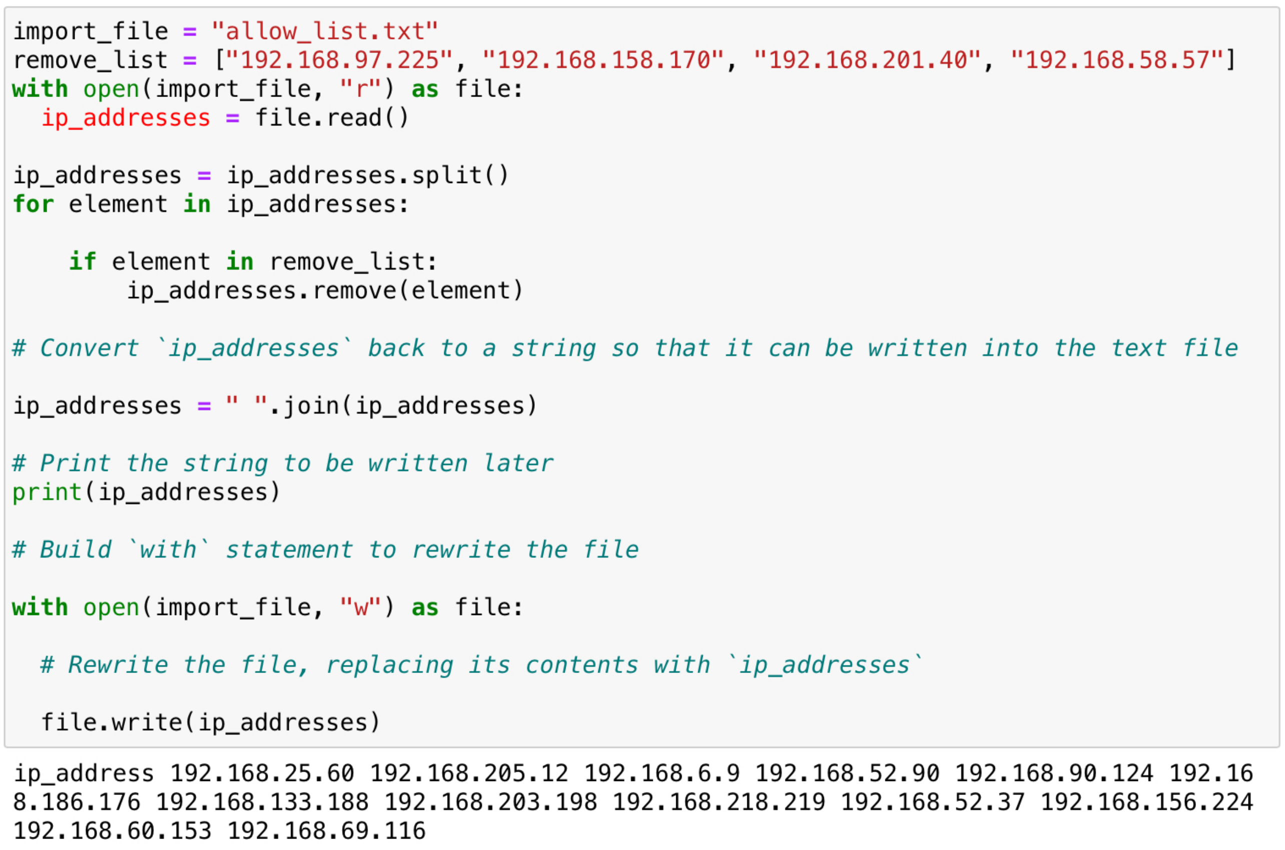
```
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
with open(import_file, "r") as file:
  ip_addresses = file.read()
ip_addresses = ip_addresses.split()

# 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)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.1
68.90.124', '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', '192.168.69.116']
```

I iterate through elements in the ip_addresses list loaded from the file allow_list.txt, and remove them from the ip_addresses list if they appear in the remove_list, using the remove() command.

Notice the printed output at the end no longer contains any IP addresses from the remove_list.

Update the file with the revised list of IP addresses



```python
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]
with open(import_file, "r") as file:
  ip_addresses = file.read()

ip_addresses = ip_addresses.split()
for element in ip_addresses:

    if element in remove_list:
        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)

# Print the string to be written later
print(ip_addresses)

# Build `with` statement to rewrite the file

with open(import_file, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.16
8.186.176 192.168.133.188 192.168.203.198 192.168.218.219 192.168.52.37 192.168.156.224
192.168.60.153 192.168.69.116
```

I combine the updated ip_addresses list into a single string, with elements in the ip_addresses list separated by the space character " " so that all the IP addresses display in the screenshot. Alternatively, I could have used the "\n" characters to indicate separating the elements by a new line.

I then overwrite the original file allow_list.txt to save the updated access list by reopening the original allow_list.txt in write mode, and writing the contents of the updated ip_addresses string to overwrite and update the file with the .write() command.

Summary

In this task I demonstrate how I explored programming in Python to update an access control list in order to protect the confidentiality of restricted files. I learned how to open and save files in python, how to convert a string into a list, update the list, search through lists and convert a list into a string.