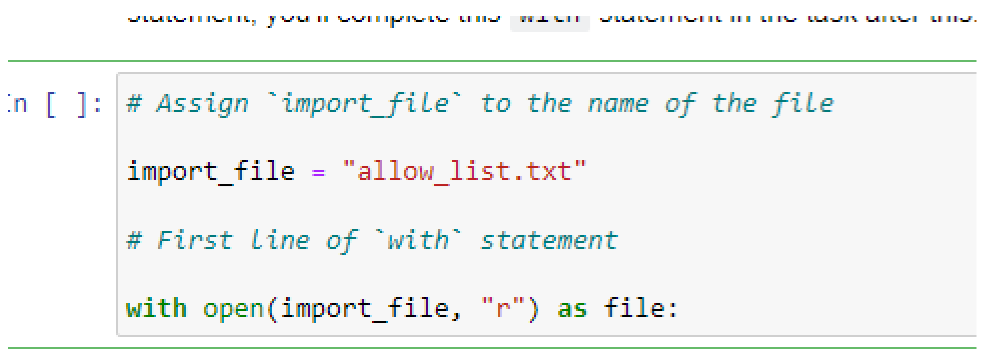
Algorithm for file updates in Python

Project description

As a security professional at a healthcare company, my task involves managing the access to restricted content by regularly updating an allow_ list of IP addresses. These IP addresses belong to employees who can access personal patient records. The goal is to ensure that any IP addresses identified on a remove list are deleted from this allow list. This project entails creating a Python algorithm to automate this task, ensuring that the allow list is consistently up-to-date and secure.

Open the file that contains the allow list

To open the file called `allow_list.txt`, I assigned the file name to a variable and used a with statement to open the file. This method ensures that the file is properly managed and closed after its contents are read.



statement, you'll complete this `with` statement in the task after this.

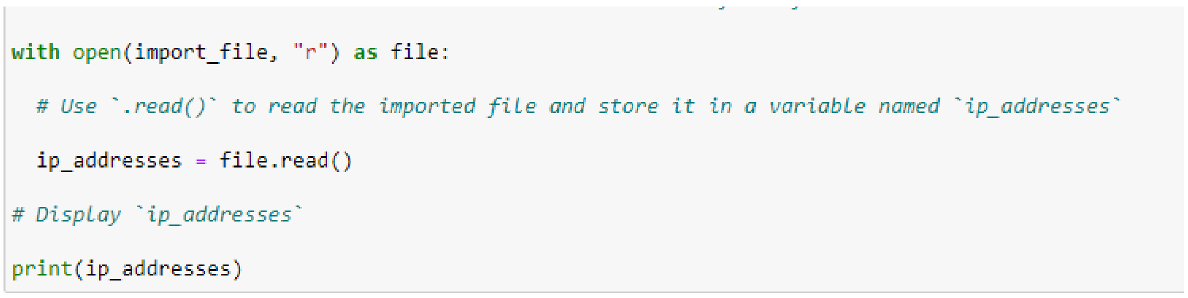
```
In [ ]: # Assign `import_file` to the name of the file

        import_file = "allow_list.txt"

        # First line of `with` statement

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

Explanation:

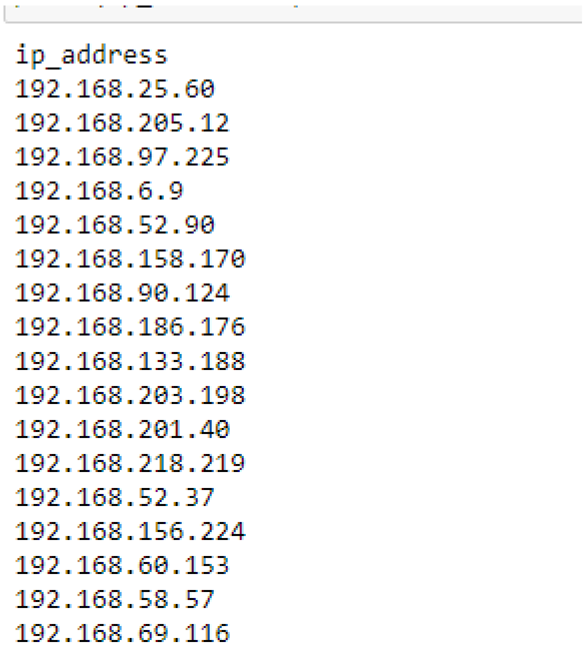
- `import_file` stores the filename.
- `with open(import_file, 'r') as file:` opens the file in read mode (`'r'`) and assigns it to the variable `file`.
- The with statement ensures that the file is automatically closed after the block of code is executed.

Read the file contents

Next, I read the contents of the file and stored them in a string variable called `ip_addresses`.



```
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)
```



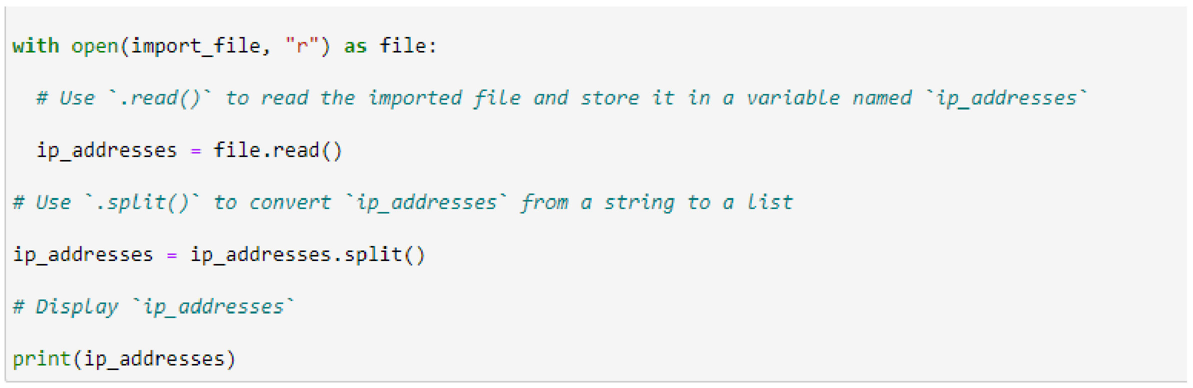
```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116
```

Explanation:

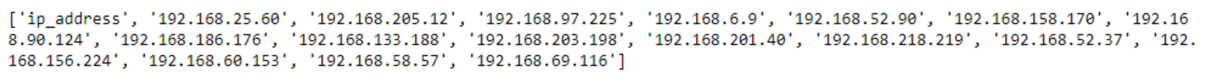
- `file.read()` reads the entire content of the file and stores it in the `ip_addresses` variable.

Convert the string into a list

To manage individual IP addresses, I converted the string of IP addresses into a list using the `.split()` method.



```
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)
```



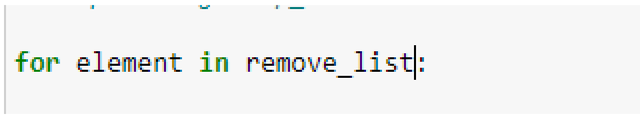
```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.170', '192.16
8.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.52.37', '192.
168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

Explanation:

- `ip_addresses.split()` splits the string at each whitespace and returns a list of IP addresses.

Iterate through the remove list

I set up a for loop to iterate through a list called `remove_list`, which contains IP addresses to be removed from the allow list.



```
for element in remove_list:
```

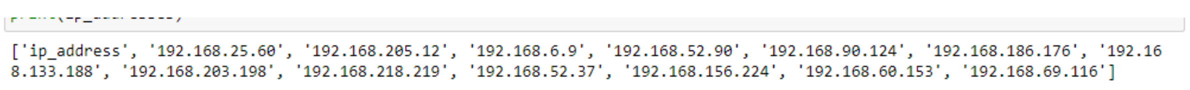
Explanation:

- `for element in remove_list:` iterates through each IP address in `remove_list`, with `element` as the loop variable.

Remove IP addresses that are on the remove list

Within the for loop, I checked if each element from the remove list was in the allow list. If so, I removed it using the `.remove()` method.

```python
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', '192.168.90.124', '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']
```

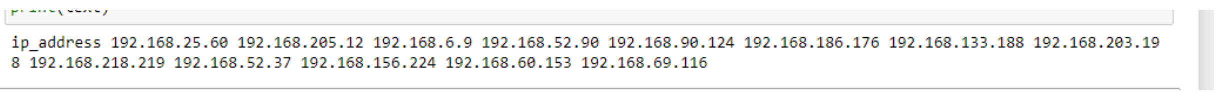
Explanation:

- `if element in ip_addresses_list:` checks if the current `element` is in the `ip_addresses_list`.
- `ip_addresses_list.remove(element)` removes the `element` from the `ip_addresses_list`.

Update the file with the revised list of IP addresses

Finally, I converted the list of IP addresses back into a string and wrote it back to the file.

```python
# 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)
```



```
ip_address 192.168.25.60 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.203.19
8 192.168.218.219 192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

Explanation:

- `'\n'.join(ip_addresses_list)` joins the list elements into a single string, separated by newline characters.
- `with open(import_file, 'w') as file:` opens the file in write mode (`'w'`).
- `file.write()` writes the updated string back to the file.

Summary

In this project, I developed a Python algorithm to manage an allow list of IP addresses for a healthcare company's restricted subnetwork. The algorithm follows these steps:

1. **Open the file**: The file is opened using a with statement and the open() function to ensure it is properly managed.
2. **Read the contents**: The .read() method converts the file contents to a string.
3. **Convert to a list**: The .split() method converts the string to a list of IP addresses.
4. **Iterate through the remove list**: A for loop checks and removes IP addresses from the allow list.
5. **Update the file**: The .join() method converts the list back to a string, and the .write() method updates the file with the revised list.