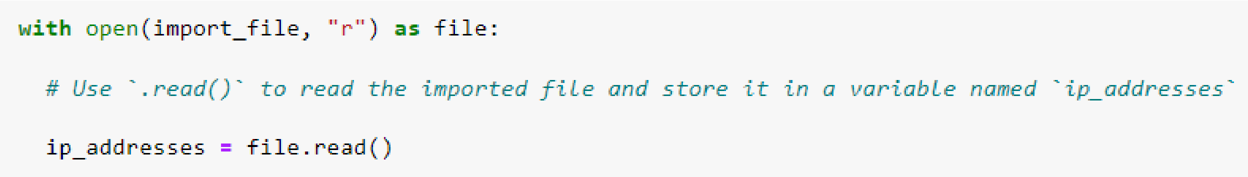
Algorithm for file updates in Python

Project description

A need arises for updating a txt file containing whitelisted IP addresses and we need to use Python to automate the task. This automation will enable increased efficiency and reliability in this repetitive process.

Open the file that contains the allow list

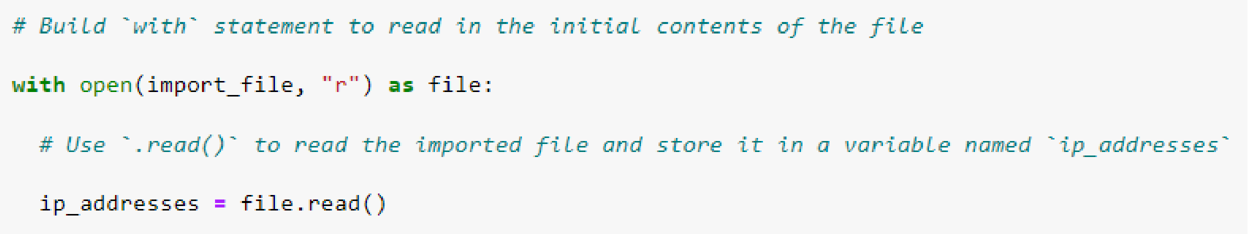
First we open the txt file read-only, “r”, so that we can extract the content. We do this cleanly by using the with() statement along with the read() method.



```
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()
```

Read the file contents

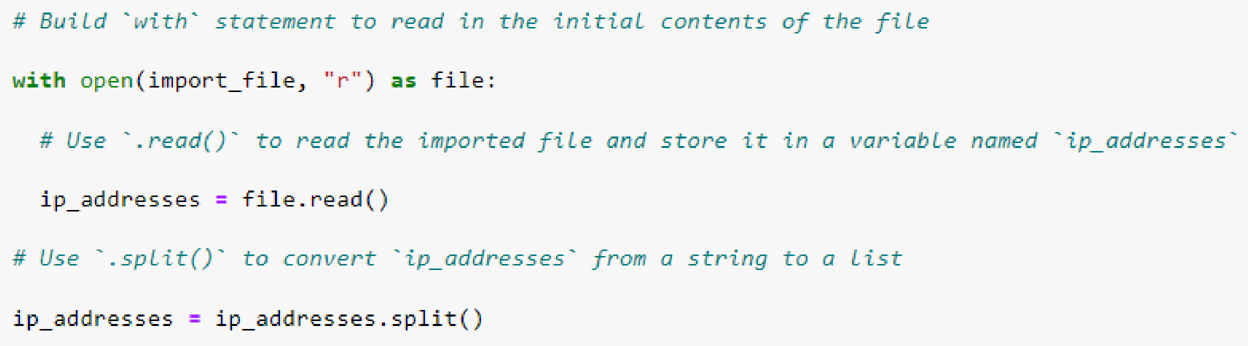
Afterwards we extract the content and assign it to a variable.



```
# 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

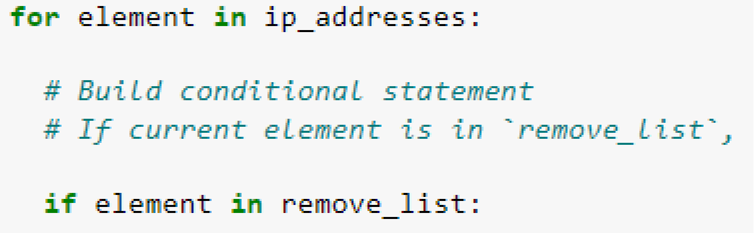
Now that we have the string assigned to a variable, we need to convert the string to an iterable list. We do this with the “split()” method, which creates the list from the string input.



```
# 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

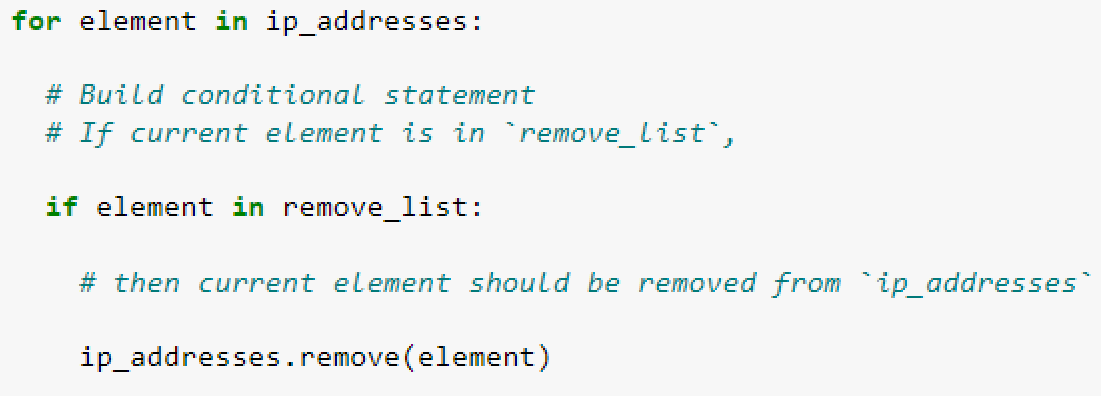
We have our iterable for the input txt file now, we’ll also need the remove_list variable. We can begin with our iterables, we do this using the for loop.



```
for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

  if element in remove_list:
```

Remove IP addresses that are on the remove list

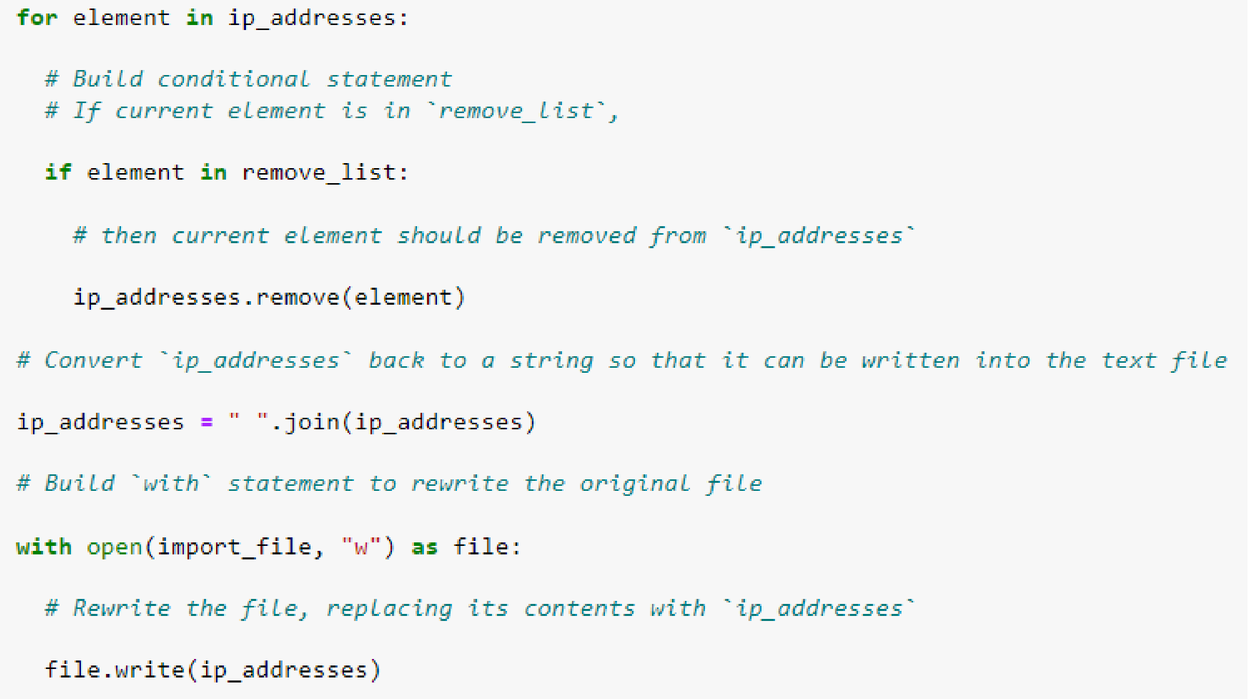
We iterate through the txt file list and remove any matches that exist within the remove_list list. When working with lists we can use the “remove()” method to essentially delete a specific line from the txt file.



```
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

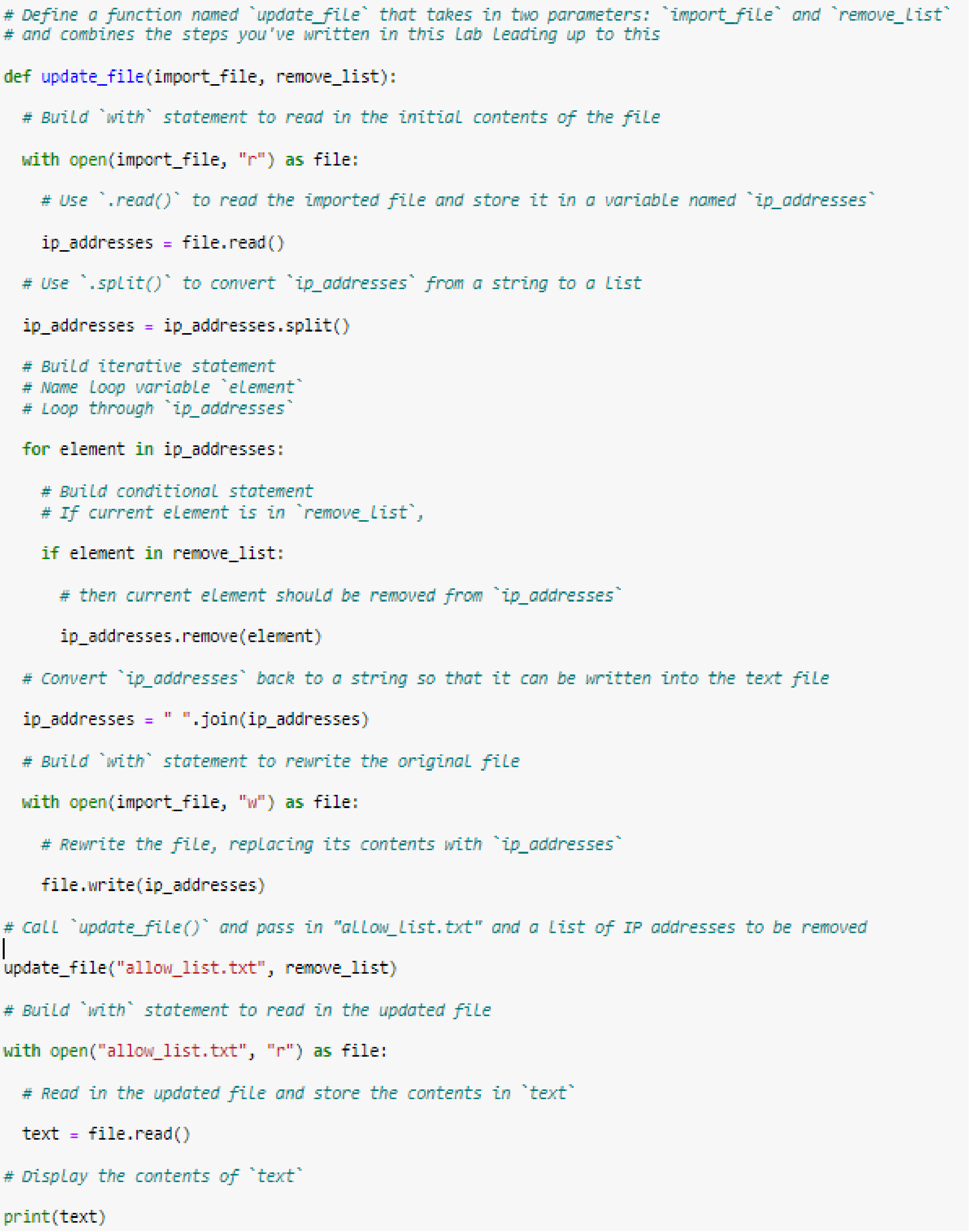
We’ve created our updated list and now need to update the original file. We’ll open it with write permissions, “w”, and overwrite its content “.write()” with the updated and string converted ip_addresses list.



```
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)

# 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

Python knowledge and experience is a very useful resource that will benefit automation of any task, including security functionalities that would otherwise require many man-hours. Creating a single function with the sections above enables easy one-line code entries throughout your app, allowing cleaner code and easier readability. Below is the complete function:



```python
# Define a function named `update_file` that takes in two parameters: `import_file` and `remove_list`
# and combines the steps you've written in this lab leading up to this

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)

# Call `update_file()` and pass in "allow_list.txt" and a list of IP addresses to be removed

update_file("allow_list.txt", remove_list)

# Build `with` statement to read in the updated file

with open("allow_list.txt", "r") as file:

  # Read in the updated file and store the contents in `text`

  text = file.read()

# Display the contents of `text`

print(text)
```