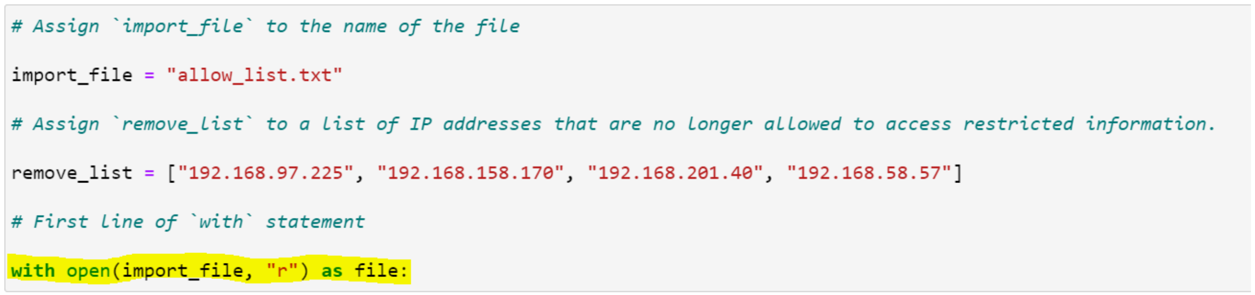
Algorithm for file updates in Python

Project description

This project details the process of creating an algorithm to open a log file to access allowed IP addresses and make changes to the file to remove IP addresses that shouldn't have access.

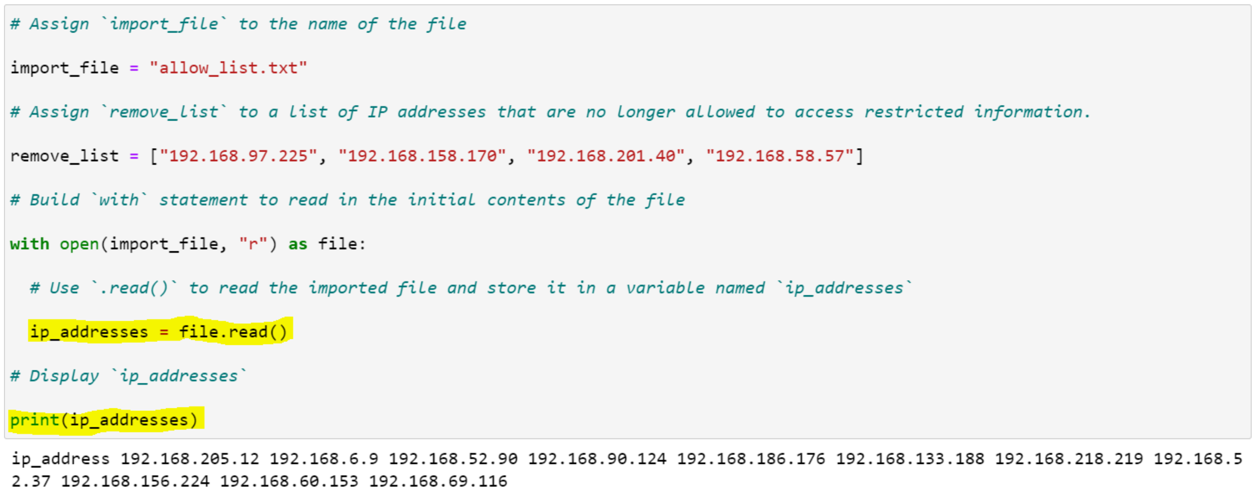
Open the file that contains the allow list



```
# 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"]

# First line of `with` statement

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

In the above code, I used "with open()" to open the file "allow_list.txt" which I stored in import_file variable earlier. The "with open()" allows us to import .txt and .csv files into Python. The "r" parameter allows Python to read the content of the file.

Read the file contents

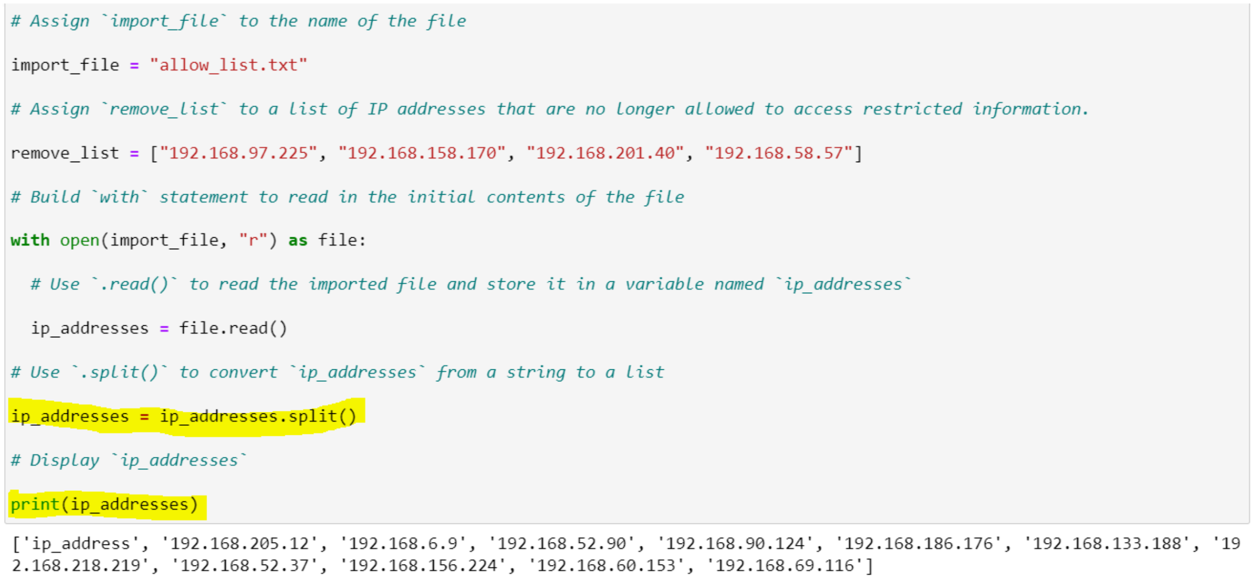


```
# Assign `import_file` to the name of the file

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)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124 192.168.186.176 192.168.133.188 192.168.218.219 192.168.5
2.37 192.168.156.224 192.168.60.153 192.168.69.116
```

In order to actually read the file, I used .read() method and store the data in a new variable called ip_addresses so even if Python closes the file after reading it, I could still access its data via ip_addresses. Then, I used the print function to display the contents of the file. The file had Ip addresses stored as a string.

Convert the string into a list

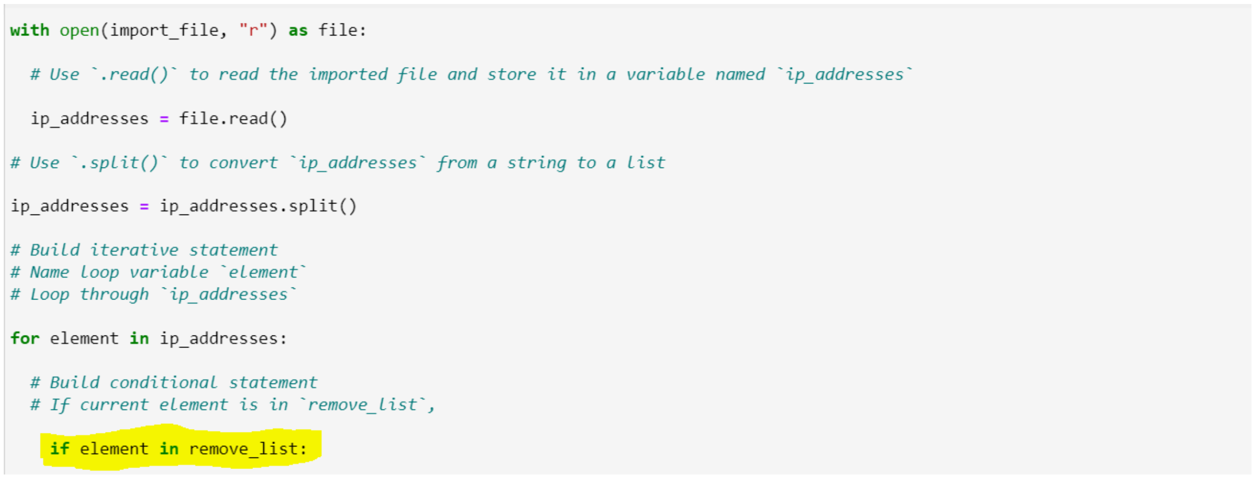


```
# Assign `import_file` to the name of the file

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', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '19
2.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

To parse the file, I had to convert the string data into a list. For this, I used the .split() method which converts string data into a list format and stored it in ip_addresses variable so I could use it later in the code.

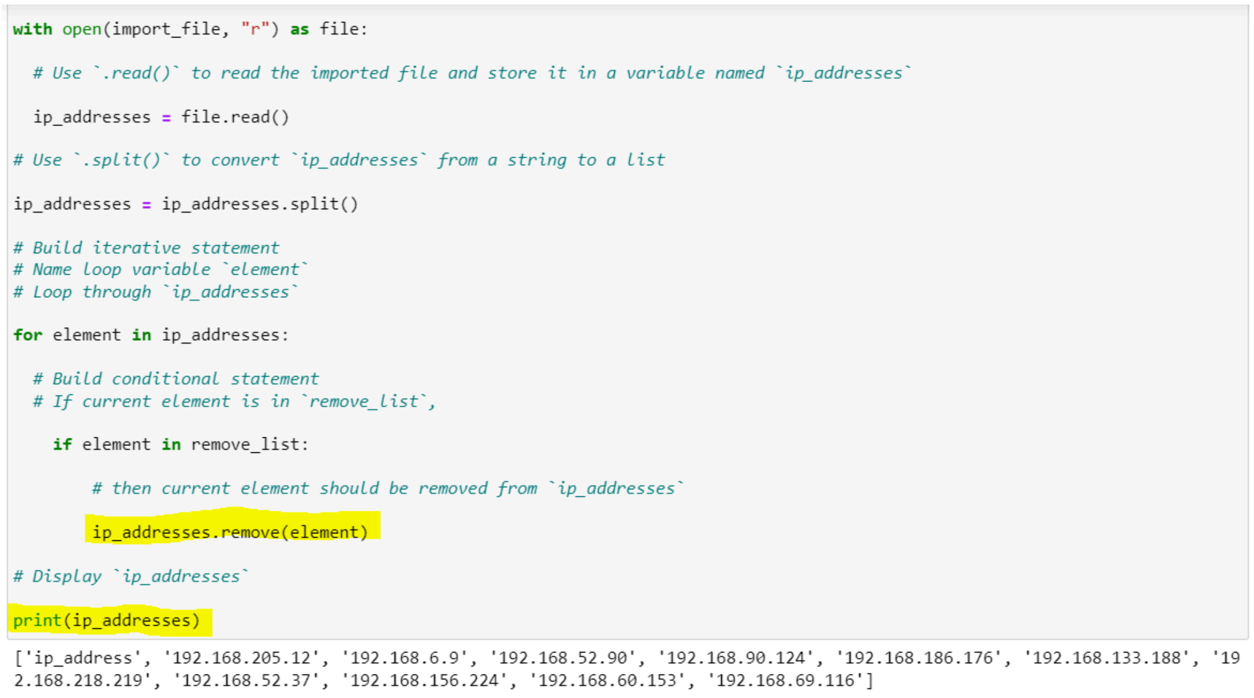
Iterate through the remove list



```
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:
```

In order to remove the IP addresses that were in remove_list variable, I had to iterate through the ip_addresses using a for loop. I used “element” as my loop variable. Then, I used if statement and use element in remove_list as the condition to look for IP addresses that were a part of remove_list.

Remove IP addresses that are on the remove list

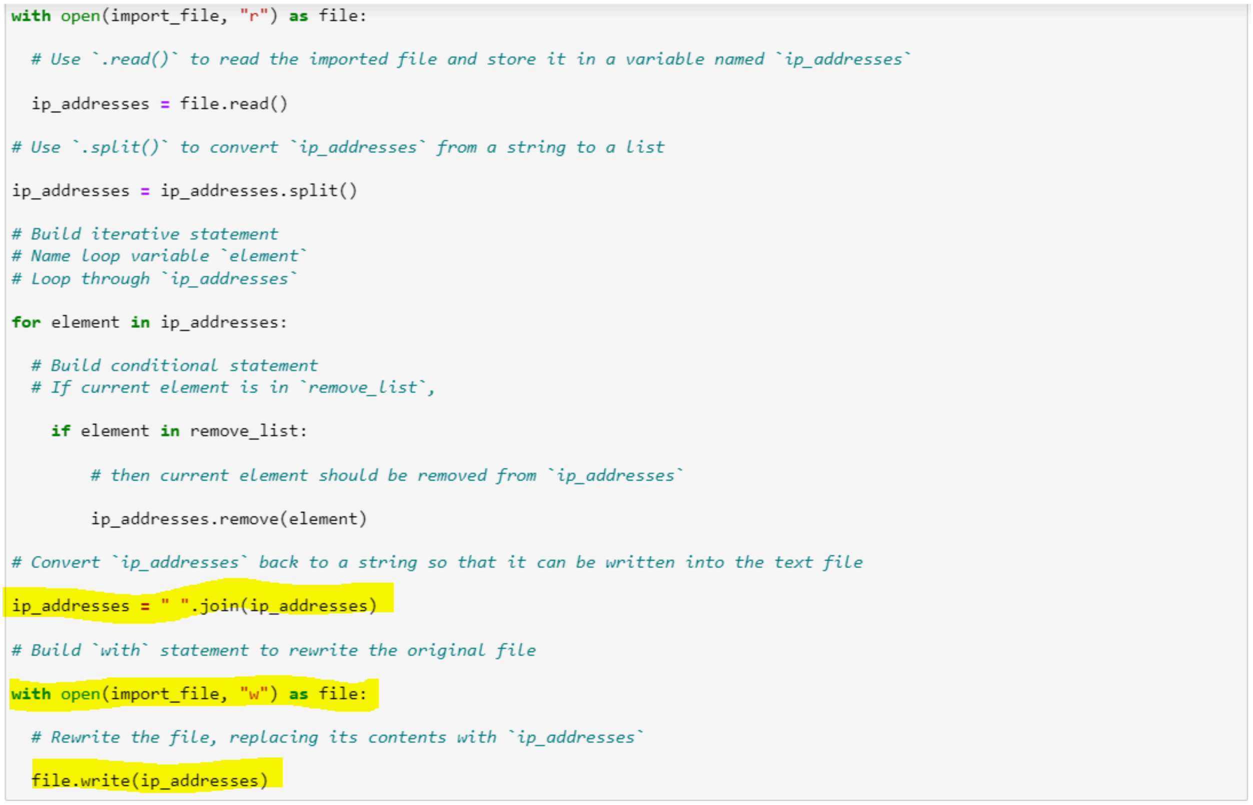


```
with open(import_file, "r") as file:

  # 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)
```

```
['ip_address', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '19
2.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

The if statement I wrote determined if the condition “elements in remove_list” is True or False. If it came out to True , it proceeds to the next line of code to remove IP addresses from ip_addresses variable that were on the remove list. I used .remove() method. I passed “element” as the argument for .remove() method.

Update the file with the revised list of 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()

# 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)
```

In this step, I updated the older file that included IP addresses that were also a part of remove_list the with newer file that excluded the IP addresses from the remove_list. First I had to convert the data from list to string data type. For this, I used .join() method which converts list data into string data type. After converting data, I used "with open()" to write the file with updated list of IP addresses stored as a string. I used "w" parameter and .write() to achieve this.

Summary

Throughout this project, I used various Python functions to create an algorithm for updating a file that contains list of allowed IP addresses. The algorithm incorporated function to import files into Python for Parsing a file. The parsing methods such as ".read() and .write()" were also a part of the algorithm to read the files and make changes to them. The for loop and if statement were also a part of the algorithm. The data conversion methods such as ".split() and .join()" were also used in the algorithm.