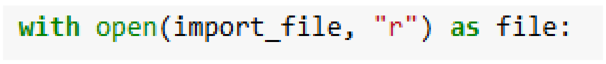
Algorithm for file updates in Python

Project description

At my organization, access to restricted systems is controlled using an allow list stored in a file called allow_list.txt. This file contains a list of IP addresses that are permitted access. Over time, access needs to be updated by removing certain IPs, whether due to policy changes, old devices or offboarding users. To make this process faster I created a Python algorithm that opens the allow list, compares it with a separate list of IP addresses that should be removed, and rewrites the file with only the approved IPs.

Open the file that contains the allow list

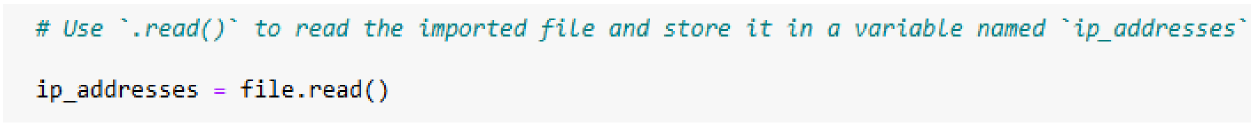
I assigned the file name (allow_list.txt) to a variable and used a with statement to open it in read mode. This ensures the file gets properly closed after reading.



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

Read the file contents

Using the .read() method, I stored all IP addresses from the file as a single string.

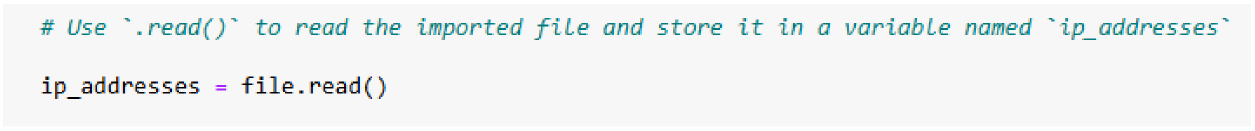


```
# Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

ip_addresses = file.read()
```

Convert the string into a list

I used the .split() method to turn the string into a list of individual IP addresses. This makes it easier to loop through and modify.



```
# Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

ip_addresses = file.read()
```

Iterate through the ip_addresses list

I created a for loop to go through each IP and compare it with a predefined remove_list.

```
# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:
```

Remove IP addresses that are on the remove list

If an IP in the list matches one from the remove_list, it gets removed using .remove().

```
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

I used .join() to convert the updated list back into a string, placing each IP on a new line. Then I opened the file again in write mode and replaced its contents with the updated list.

```
# 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

This algorithm makes it easy to keep the allow list up to date by automatically removing IP addresses that should no longer have access. It reads the file, checks each IP against a remove list, and then rewrites the file with only the approved addresses. This saves time, reduces the chance of mistakes, and helps make sure access control stays accurate.