

Algorithm for file updates in Python

Project description

At my organization, access to restricted content is controlled with an allow list of IP addresses. The "allow_list.txt" file identifies these IP addresses. A separate remove list identifies IP addresses that should no longer have access to this content. I created an algorithm to automate updating the "allow_list.txt" file and remove these IP addresses that should no longer have access.

Open the file that contains the allow list

For the first part of the algorithm, I opened the "allow_list.txt" file. First, I assigned this file name as a string to the `import_file` variable:

```
import_file = "allow_list.txt"
```

Then, I used a `with` statement to open the file:

```
#Build with the statement to read in the initial content of the file
```

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

The `with` statement, combined with `.open()`, allows the file to be opened safely in read mode ("r") and ensures it is automatically closed when the block completes. The `as file` syntax assigns the file object to the variable `file` for use within the block.

Read the file contents

In order to read the file contents, I used the `.read()` method to convert it into the string.

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

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

```
ip_address = file.read()
```

The `.read()` method converts the file contents into a single string, which allows me to process and manipulate the data within Python. This string is assigned to the variable `ip_addresses`.

In summary, this code reads the contents of the "allow_list.txt" file into a string format that allows me to later use the string to organize and extract data in my Python program.

Convert the string into a list

In order to remove individual IP addresses from the allow list, I needed it to be in list format. Therefore, I next used the `.split()` method to convert the `ip_addresses` string into a list:

```
#use .split to convert ip_address from string to list
```

```
Ip_address = ip_address.split()
```

By default, `.split()` separates the string into elements based on whitespace. This converts the string of IP addresses into a list stored in `ip_addresses`, making it easier to remove elements. Iterate through the remove list

A key part of my algorithm involves iterating through the IP addresses that are elements in the `remove_list`. To do this, I incorporated a for loop:

```
#Create iterative statement , Name loop variable element, loop through the remove list
```

```
for element in remove_list:
```

The for loop in Python repeats code for a specified sequence. The overall purpose of the for loop in a Python algorithm like this is to apply specific code statements to all elements in a sequence. The `for` keyword starts the for loop. It is followed by the loop variable element and the keyword `in`. The keyword `in` indicates to iterate through the sequence `ip_addresses` and assign each value to the loop variable element.

Remove IP addresses that are on the remove list

My algorithm requires removing any IP address from the allow list, `ip_addresses`, that is also contained in `remove_list`. Because there were not any duplicates in `ip_addresses`, I was able to use the following code to do this:

For element in remove_list:

#Create condition statement to evaluate if element is in ip_address

if element in ip_address:

#use the. remove() method to remove

#element from ip_address

ip_address.remove(element)

This conditional prevents errors that would occur if .remove() were called on an element not present in the list. Any IP address found in both remove_list and ip_addresses is removed.

Then, within that conditional, I applied .remove() to ip_addresses. I passed in the loop variable element as the argument so that each IP address that was in the remove_list would be removed from ip_addresses.

Update the file with the revised list of IP addresses

As a final step in my algorithm, I needed to update the allow list file with the revised list of IP addresses. To do so, I first needed to convert the list back into a string. I used the .join() method for this:

#convert ip_address back to a string so that it can be written into the text file

Ip_address = "\n".join(ip_address)

The .join() method combines all items in an iterable into a string. The .join() method is applied to a string containing characters that will separate the elements in the iterable once joined into a string. In this algorithm, I used the .join() method to create a string from the list ip_addresses so that I could pass it in as an argument to the .write() method when writing to the file "allow_list.txt". I used the string ("\n") as the separator to instruct Python to place each element on a new line.

Then, I used another with statement and the .write() method to update the file:

#Build with statement to rewrite the original file

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

```
#Rewrite the file replacing its content with ip_addresses
```

```
file.write(ip_addresses)
```

This time, I used a second argument of "w" with the open() function in my with statement. This argument indicates that I want to open a file to write over its contents. When using this argument "w", I can call the .write() function in the body of the with statement. The .write() function writes string data to a specified file and replaces any existing file content.

In this case I wanted to write the updated allow list as a string to the file "allow_list.txt". This way, the restricted content will no longer be accessible to any IP addresses that were removed from the allow list. To rewrite the file, I appended the .write() function to the file object file that I identified in the with statement. I passed in the ip_addresses variable as the argument to specify that the contents of the file specified in the with statement should be replaced with the data in this variable.

Summary

I developed an algorithm to automate updating the "allow_list.txt" file by removing IP addresses listed in a remove_list. The algorithm:

- Opens the allow list file in read mode.
- Reads its contents and converts it into a string.
- Splits the string into a list of IP addresses (ip_addresses).
- Iterates through remove_list, removing any IP addresses found in ip_addresses.
- Converts the updated list back into a string and writes it to "allow_list.txt".

This approach ensures that only authorized IP addresses remain in the allow list, efficiently managing access to restricted content.