Python Algorithm for File Updates

Updating a File through a Python Algorithm

Wendy Alayon
26/08/2024 **@syszern**

Algorithm for File Updates in Python

Scenario

A security professional at a healthcare company is tasked with regularly updating a file that lists employees authorized to access restricted content. This file is based on those working with personal patient records, with access restrictions determined by IP address. An allow list specifies the IP addresses permitted to sign into the restricted subnetwork, while a remove list identifies IP addresses that need to be removed from this allow list.

The task is to create a Python algorithm to check if any IP addresses on the remove list are present in the allow list. If such IP addresses are found, they should be removed from the allow list file.

Project description

In the organization, access to restricted content is managed using an allow list of IP addresses. The file "`allow_list.txt`" contains these IP addresses. Additionally, a separate remove list specifies IP addresses that should no longer have access. An algorithm has been developed to automate the process of updating the "`allow_list.txt`" file by removing the IP addresses listed on the remove list.

Open the file that contains the allow list

The file to be opened is named `"allow_list.txt"`. Assign this file name as a string to the `import_file` variable.

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

import_file = "allow_list.txt"
```

Next, use a `with` statement to open the file. Store the file in the variable `file` while working with it within the `with` statement.

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

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

The `with` statement is used with the `.open()` function in read mode to open the allow list file for reading. This approach provides access to the IP addresses stored in the file. The `with` keyword helps manage resources by ensuring the file is closed after exiting the with statement. In the code `with open(import_file, "r") as file:`, the `open()` function takes two parameters: the first specifies the file to open, and the second indicates the mode, with `"r"` signifying read mode. The `as` keyword is used to assign the variable `file`, which stores the file object returned by the `.open()` function while within the `with` statement.

Read the file contents

Next, use the `.read()` method to convert the contents of the allow list file into a string for reading. Store this string in a variable named `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()
```

When using the `.open()` function with the `"r"` argument for "read," the `.read()` method can be called within the with statement. The `.read()` method converts the file contents into a string. Applying the `.read()` method to the `file` variable specified in the with statement stores the string output in the variable `ip_addresses`.

This code reads the contents of the `"allow_list.txt"` file into a string format, enabling the subsequent organization and extraction of data within the Python program.

Convert the string into a list

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, use the `.split()` method to convert the `ip_addresses` string into a list.

```
# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()
```

The `.split()` function is used by appending it to a string variable. It converts the contents of the string into a list. The function is used to split `ip_addresses` into a list to facilitate the removal of IP addresses from the allow list. By default, `.split()` divides the

text by whitespace into list elements. In this process, `.split()` operates on the data in the `ip_addresses` variable, which contains a string of IP addresses separated by whitespace, and converts this string into a list of IP addresses. The resulting list is then reassigned to the `ip_addresses` variable.

Iterate through the remove list

A second list called `remove_list` contains all of the IP addresses that should be removed from the `ip_addresses` list. Set up the header of a for loop that will iterate through the `remove_list`. Use element as the `loop` variable.

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

for element in remove_list:
```

The `for` loop in Python executes code repeatedly for a specified sequence. Its primary function is to apply specific code statements to each element within the sequence. The `for` keyword initiates the loop, followed by the loop variable `element` and the keyword in. The in keyword directs the loop to iterate through the sequence `remove_list` and assigns each value to the loop variable `element`.

Remove IP addresses that are on the remove list

In the body of the iterative statement, include code to remove all IP addresses from the allow list that are also present on the remove list. Begin by creating a conditional statement to check if the loop variable `element` is part of the `ip_addresses` list. Within that conditional, use the `.remove()` method on the `ip_addresses` list to eliminate the IP addresses identified by the loop variable `element`.

```
for element in remove_list:

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

    if element in ip_addresses:

        # use the `.remove method` to remove elements from `ip_addresses`

        ip_addresses.remove(element)
```

First, within the `for` loop, a conditional was created to evaluate whether the loop variable `element` was present in the `ip_addresses` list. This check was necessary because applying `.remove()` to elements not found in `ip_addresses` would cause an error.

Within the conditional, the `.remove()` method was used on `ip_addresses`, with the loop variable `element` passed as the argument to remove each IP address present in the `remove_list` from `ip_addresses`.

Update the file with the revised list of IP addresses

After removing the IP addresses from the `ip_addresses` variable, the algorithm can be completed by updating the file with the revised list. First, convert the `ip_addresses` list back into a string using the `.join()` method, applying `.join()` to the string `"\n"` to separate the elements by placing each on a new line.

Next, use another `with` statement and the `.write()` method to overwrite the file assigned to the `import_file` variable.

```
ip_addresses = "\n".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)
```

The `.join()` method combines all items in an iterable into a string, using a specified separator. In this algorithm, the `.join()` method was used to create a string from the `ip_addresses` list, with `"\n"` as the separator to place each element on a new line.

To update the file, another `with` statement and the `.write()` method were used. The `open()` function was called with the argument `"w"`, which indicates that the file should be opened for writing, replacing any existing content. The `.write()` method was then used to write the updated list to the file `"allow_list.txt"`. This ensured that the file content was replaced with the updated `ip_addresses` list, removing any IP addresses that were no longer to be included.

Summary

An algorithm was developed to remove IP addresses listed in the remove_list variable from the `"allow_list.txt"` file of approved IP addresses. The algorithm opens the file, reads its content into a string, and converts this string into a list stored in the variable ip_addresses. It then iterates through the IP addresses in `remove_list`, checking if each element is present in `ip_addresses`. If an element is found, the `.remove()` method is applied to remove it from `ip_addresses`. Finally, the `.join()` method is used to convert

`ip_addresses` back into a string, which is then written to overwrite the contents of the `"allow_list.txt"` file with the updated list.