

Algorithm for file updates in Python

Project description

In this project, I developed an automated solution to manage access to restricted content by maintaining an allowed list of IP addresses. The project centers around two key files: "allow_list.txt", which stores IP addresses with permission to access the content, and a separate remove list containing IP addresses that should be revoked from access. My algorithm automates the process of updating the "allow_list.txt" file by removing IP addresses that are no longer authorized.

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:
```

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. In my algorithm, the `with` statement is used with the `.open()` function in read mode to open the allow list file for the purpose of reading it. The purpose of opening the file is to allow me to access the IP addresses stored in the allow list file. The `with` keyword will help manage the resources by closing the file after exiting the `with` statement. In the code `with open(import_file, "r") as file:`, the `open()` function has two parameters. The first identifies the file to import, and then the second indicates what I want to do with the file. In this case, "r" indicates that I want to read it. The code also uses the `as` keyword to assign a variable named `file`; `file` stores the output of the `.open()` function while I work within the `with` statement.

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)

```

When using an `.open()` function that includes the argument "r" for "read," I can call the `.read()` function in the body of the `with` statement. The `.read()` method converts the file into a string and allows me to read it. I applied the `.read()` method to the file variable identified in the `with` statement. Then, I assigned the string output of this method 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.

Output:

```

ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
192.168.52.90
192.168.158.170
192.168.90.124
192.168.186.176
192.168.133.188
192.168.203.198
192.168.201.40
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.58.57
192.168.69.116

```

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)
```

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. The `.split()` function is called by appending it to a string variable. It works by converting the contents of a string to a list. The purpose of splitting `ip_addresses` into a list is to make it easier to remove IP addresses from the allow list. By default, the `.split()` function splits the text by whitespace into list elements. In this algorithm, the `.split()` function takes the data stored in the variable `ip_addresses`, which is a string of IP addresses that are each separated by a whitespace, and it converts this string into a list of IP addresses. To store this list, I reassigned it back to the variable `ip_addresses`.

Output:

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.97.225', '192.168.6.9', '192.168.52.90', '192.168.158.170', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

Iterate through the remove 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()

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

    # Display `element` in every iteration
    print(element)
```

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. 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

```
# 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()

# 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)
```

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 above code to do this.

First, within my for loop, I created a conditional that evaluated whether or not the loop variable `element` was found in the `ip_addresses` list. I did this because applying `.remove()` to elements that were not found in `ip_addresses` would result in an error.

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`.

Output:

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

Update the file with the revised list of IP addresses

```
# 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()

# 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 = "\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)
```

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. 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.

Output:

```
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
192.168.52.37 192.168.156.224 192.168.60.153 192.168.69.116
```

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. 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.

Creating and Calling a function to update the file

In this step, the function `update_file()` is defined to automate the process of removing unauthorized IP addresses from a file. It takes two parameters: `import_file`, the file containing allowed IPs, and `remove_list`, the list of IPs to be removed. The function first reads the contents of the file and converts the string of IP addresses into a list. It then checks each IP in the list, removing any that match those in the `remove_list`. After making the necessary updates, the list is converted back into a string and written over the original file. The function is then called with the `"allow_list.txt"` file and a list of IPs to be removed. Finally, the updated file is read and displayed to verify the changes, ensuring that only the allowed IP addresses remain.

```
# 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", ["192.168.25.60", "192.168.140.81", "192.168.203.198"])  
# 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)
```


Summary

I created an algorithm that removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of approved IP addresses. This algorithm involved opening the file, converting it to a string to be read, and then converting this string to a list stored in the variable `ip_addresses`. I then iterated through the IP addresses in `remove_list`. With each iteration, I evaluated if the element was part of the `ip_addresses` list. If it was, I applied the `.remove()` method to it to remove the element from `ip_addresses`. After this, I used the `.join()` method to convert the `ip_addresses` back into a string so that I could write over the contents of the `"allow_list.txt"` file with the revised list of IP addresses.

((This project demonstrates an automated approach to managing access control by removing specific IP addresses from an allowed list. The algorithm efficiently updates the `allow_list.txt` file by first converting its contents to a string, then transforming it into a list of IP addresses. It iterates through a `remove_list` of IPs, checks for matches in the allowed list, and removes any found IPs using the `.remove()` method. Finally, the updated list is converted back into a string and written over the original `allow_list.txt` file, ensuring that access restrictions are kept up to date without manual intervention. This solution can easily be adapted for similar scenarios requiring automated file updates for managing access or other resources.))