Algorithm for file updates in Python

Project description

As part of my job, I am required to regularly update a file that identifies the employees who can access restricted content. Employees are restricted access based on their IP address stored in the file `allow_list.txt`. Any IP addresses that should no longer have access should be removed. I have decided to create a Python algorithm to automate the task instead of updating the file manually which could be very time consuming. This will give me time to work on other tasks that cannot be automated as easily.

Open the file that contains the allow list

Firstly, I want to open the file that contains the list of allowed IP addresses that can access the restricted files. To do this, I used the below code.

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

import_file = "allow_list.txt"
```

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

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

The `with` statement is used to close the file automatically after exiting the `with` statement.

The `open()` function is used to open a file in Python. The first argument that's passed to it is the file to be opened. In this case I have stored the file as a variable called `import_file` so it can be found easily near the top and changed quickly. The variable is holding the string `"allow_list.txt"` which is the name of the file that needs opening. Since the file is in the same location as this script, I do not need to use a file path. The second argument determines what to do with the file. I have passed the argument `"r"` which tells Python that I want to read the file.

The `as` keyword is used to assign a variable that references the file, in this case the variable is `file`.

Read the file contents

Next, I want to store the content of the file as a string variable so that I can perform string operations on it. To do this, I used the below code.

```
# 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()
```

The `.read()` method is used to convert files into strings. This is necessary in order to use and display the contents of the file that was read.

I then stored the read content as the variable `ip_addresses` to be used later.

Convert the string into a list

Next, I want to convert the variable from a string to a list. This is so that I can iterate through the list and easily operate with each of the ip addresses from the file. To do this, I used the below code.

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

ip_addresses = ip_addresses.split()
```

The `.split()` method is used to convert a string into a list. The method does this by separating the string based on the specified character passed to it as an argument. If no argument is passed, it defaults to whitespace. In this case, the file is stored with new line characters separating the ip addresses, this is considered a whitespace in Python so I will pass no arguments.

I then stored it back as itself so I don't have to create a new variable to hold the list.

Iterate through the remove list

Next, I want to iterate through the list of IP addresses that need to be removed from `allow_list`. The IP addresses that need to be removed are in a list variable `remove_list` that has been defined prior. To do this I used the below code.

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

for element in remove_list:
```

The statement `for` is used to create a `for` loop. `for` loop is used to repeat code for a specified sequence. In this case the sequence is `remove_list`, the `for` loop will iterate through each value in the `remove_list` list.

The variable `element` is the reference that holds the value in `remove_list` for the current iteration of the loop.

The `in` keyword is used to determine whether a given value is an element of a sequence. In this case, the sequence is `remove_list`.

Remove IP addresses that are on the remove list

Next, I want to check if any of the IP addresses in `remove_list` is in `ip_addresses` list. If it is then I will remove it from the list. To do this, I used the below code.

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

for element in remove_list:

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

    if element in ip_addresses:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)
```

The conditional statement, `if`, is used to test a condition. In this case it is combined with the keyword `in` to check if the value of the `element` variable is also a value in `ip_addresses`.

If the condition is true, the code indented below it, `ip_addresses.remove(element)` will execute.

The `.remove()` method is used to remove the first occurrence of a specific element in a list. This is determined by the value passed as the method's argument. In this case the method will remove the value of the variable `element` from `ip_addresses`.

This will continue until the loop iterates through all the elements in `remove_list`.

Update the file with the revised list of IP addresses

Lastly, after all the IP addresses that need to be removed have been removed from the list, I want to update the file `allow_list` with the new list. To do this, I used the below code.

```
# 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)
```

Firstly, I convert the list back to string to be stored in the file. I used the method `.join()` to concatenate the elements in the list together. The method is applied to the string that is used to separate each element in the list, in this case I want them all to be on their own line so the string used is the new line character `“\n”`. The argument passed into the method is the list that I want to convert to string. In this case, it is the `ip_addresses` list. The string is stored as itself again so no new variable needs to be created.

Next, I opened the file again with a similar command to when I wanted to read the file to use its content earlier `with open(import_file, “w”) as file`. The main difference is that the second argument is `“w”` instead of `“r”`. This tells python that I want to write to the file with my new string. This string will replace the current content of the file.

The `.write()` method is then used to write the string `ip_addresses` to the file. The argument is the string that needs writing to the file.

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.

First, it opens a file and reads its content, and then stores the content as a variable. It then converts the variable into a list and updates its content by iterating through the list and removing any IP addresses that should no longer have access from the `remove_list`. After the content is updated, the variable is then converted back into string and the file is then overwritten with the new string. This saves a lot of manual effort and time as I can just pass in the new IP addresses to be removed and execute the program again to update the list of allowed IP addresses.