Update a file through a Python algorithm

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

First, I assigned the string value `"allow_list.txt"` to the variable `import_file`, which enables me to refer to the file as `import_file` later in the program:

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

import_file = "allow_list.txt"
```

Then, I opened the allow list file using a `with` statement:

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

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

Here, `with` is used with the `open()` function to allow me to access the IP addresses stored in the allow list file. The `open()` function receives two arguments. The first argument defines the file to open. The second argument defines what to do with the file. Since I am only required to read the file contents at this point, I use `"r"` for read. The `as` keyword assigns a variable named `file`, which will be used to store the output of the `open()` function.

Read the file contents

In order to read the file contents, I first converted the file contents into a string using the `.read()` method:

```
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()
```

This code uses the `read()` method to read the `file` and assigns the resulting string output to a new variable `ip_addresses`.

Convert the string into a list

In order to better manage the IP addresses, I need to convert the string into a list format. Therefore I used the `.split()` method:

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

ip_addresses = ip_addresses.split()
```

The `.split()` method works by splitting up a string of text into a list, via detection of a whitespace. This code uses the `.split()` method to split the string of IP addresses stored in the variable `ip_addresses`, into a list of IP addresses and reassigns it back to `ip_addresses`.

Iterate through the remove list

I need to set up a `for` loop so that the algorithm can check through each and every IP address in the `remove_list`:

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

for element in remove_list:
```

The `for` loop begins with the `for` keyword. It is followed by defining the loop variable `element` and the keyword `in`. The keyword `in` will get the algorithm to iterate through `remove_list` and as the loop runs, the variable `element` takes on the value of each item in `remove_list` sequentially.

Remove IP addresses that are on the remove list

In order to check for and remove IP addresses on `remove_list`, I need to check each and every `element` against the allow list (`ip_addresses`). I do this by adding an `if` conditional within the previously constructed `for` loop:

```
for element in remove_list:

  # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

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

        ip_addresses.remove(element)
```

The `if` conditional checks whether an `element` from the `remove_list` was found in the `ip_addresses` list. If there is a match, the `.remove()` method will be applied to `ip_addresses` to remove that `element` from it.

Do note that this only works because there were no duplicates in `ip_addresses`, If there are duplicates in `ip_addresses` then the `for` loop will only remove the first instance of a matching `element`.

Update the file with the revised list of IP addresses

Last but not least, `allow_list.txt` needs to be updated with the revised list of IP addresses, less removed IP addresses. To convert the list back into a string. I used the `.join()` method:

```
# Convert `ip_addresses` back to a string so that it can be written into the text file

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

This creates a string from the list `ip_addresses` so that I could pass it in as an argument to the `.write()` method later on 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:

```python
# 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)
```

Since I am required to write to the file this time, the second argument of the `open()` function is `"w"`. This allows me to 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.

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

This algorithm removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of whitelisted IP addresses. The algorithm opens the file, converts its contents to a string to be read, and converts this string to a list stored in the variable `ip_addresses`. It then iterates through the IP addresses in `remove_list`. With each iteration, the algorithm checks if the element is also a part of the `ip_addresses` list. If yes, it proceeds to apply the `.remove()` method to remove the element from `ip_addresses`. Lastly, it uses the `.join()` method to convert the `ip_addresses` back into a string and overwrites the contents of the `"allow_list.txt"` file with the revised list of IP addresses.