

Algorithm for file updates in Python

Project description

I am on the security team of a healthcare company. My task is to use Python to create an algorithm that updates an allow list file containing IP addresses of employees that have access to a restricted subnetwork. I will build the algorithm to remove IP addresses in the remove list from the allow list and update the file containing the allow list.

Open the file that contains the allow list

First, I assign a string containing the name of the allow list file to the variable `import_file`. To open the file, I use a `with` statement. The `with` keyword is used to automatically close the file once it is done being read. After `with`, I use the `open()` function, which opens a file. The two parameters that the `open()` function takes are the name of the file to be opened and what Python will do with the file. In this case, the name of the file is in the `import_file` variable, and I use `"r"` to tell Python to read the file. I use the `as` keyword to assign the contents of the file to the variable `file`, which can be used within the `with` statement. The header of the `with` statement ends with a colon.

```
# Assign `import_file` to the name of the file  
import_file = "allow_list.txt"  
  
# First line of `with` statement  
with open(import_file, "r") as file:
```

Read the file contents

The `.read()` method converts files into strings. I place the method at the end of the file that I want to convert, which is in the `file` variable. I assign this string to the variable `ip_addresses`. This line of code is indented because it is part of the `with` statement.

```
# 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()
```

Convert the string into a list

The `.split()` method converts strings into lists. Since there is no argument passed into the method, it will separate the string wherever there is a blank space. Each part of the string will be an element in the list. The method is placed at the end of the string that I want to convert into a list. I reassign this list to the variable `ip_addresses`.

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

Iterate through the remove list

The list of IP addresses to remove from the allow list is contained in the variable `remove_list`. To iterate through the remove list, I write a `for` loop. A `for` loop is used to execute code on each element of a sequence. This statement begins with the keyword `for`. Next is the loop variable, which I named `element`. Each element of the list will be assigned to the loop variable as the list is iterated through. After the loop variable, is the keyword `in` and the name of the list to iterate through. The header of the iterative statement ends with a colon.

```
# Build iterative statement to loop through remove_list  
# Name loop variable 'element'  
  
for element in remove_list:
```

Remove IP addresses that are on the remove list

To remove the IP addresses on the remove list from the `ip_addresses` list, I iterate through the remove list and write a conditional statement within the iterative statement. To build the conditional statement, I start with the keyword `if`. I use the operator `in` to check if the element from `remove_list` is in `ip_addresses`. The header of the conditional statement ends with a colon. The body of the conditional statement is indented and tells Python what to do if the condition is `True`. I use the `.remove()` method, which removes the first occurrence of an element from a list. The method is attached to the end of the list and the element to

remove is passed through the method as an argument. The code works in this case because there are no repeats in the `ip_addresses` list.

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

for element in remove_list:

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

        if element in ip_addresses:

            # then current element should be removed from `ip_addresses`

            ip_addresses.remove(element)
```

Update the file with the revised list of IP addresses

To update the file with the new list of IP addresses, I first convert the list of IP addresses back into a string and reassign the `ip_addresses` variable with this string. The `.join()` method converts lists to strings. The method is attached to a string of characters that will connect each element of the list. In my code, I used the string `"\n"` so each element of the list would be a new line in the string. The list that will be converted into a string is the argument for the `.join()` method. To update the file, I open the file in `import_file` as I did earlier, but I use `"w"` instead of `"r"`. This allows me to overwrite the existing file. In the body of the `with` statement, I use the `.write()` method, which writes string data on a specified line. The method is attached to the end of the file to write to and the argument is the string data to write.

```
# 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)
```

Summary

The steps of building this algorithm was to first open and read the file containing the allow list. I used the `.split()` method to convert the string data of the allow list into an iterable list. I then wrote a for loop to iterate through the remove list. Within the iterative statement, I wrote a conditional statement to check if the element in the remove list appears in the list of allowed IP addresses. If the element is in the allow list, I used the `.remove()` method to remove the element from the allow list. I used the `.join()` method to convert the list back into a string, which can then be written to the allow list file.

This lab puts into practice the use of loops and conditional statements to build a Python algorithm. It also explores a way to manipulate string data in order to work with it in a loop. The simple algorithm makes it much easier and more efficient to check the allow list and remove the necessary IP addresses.