Algorithm for File Updates in Python

Completed by: Hope Tan (github.com/hope-tan)

Project description

In this scenario, I am a security professional at a fake health care company. My role includes responsibility to track which employees can access restricted content, based off of their IP address. This means that I allow or block IP addresses, depending on the employee's authorization.

To increase efficiency, I decided to write a script in Python to automate this task. I used Jupyter Notebook to deploy the code cells.

I documented the step-by-step process I took to write a Python script that **automates white and blacklisting of IP addresses for restricted content access.** This script modifies the list of allowed IP addresses by automatically removing the IP addresses that appear in a remove list from the allow_list.txt file.

Open the file that contains the allow list

First, I assigned an "import_file" variable to hold the filename of "allow_list.txt".

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

import_file = "allow_list.txt"
```

This line opens the file. "with" tells the program to include the file. "open()" opens the file and describes its use case (in this case, "r" for read). "file" is the name of the variable that holds the file contents.

```
# First line of `with` statement

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

Read the file contents

Next, I leveraged ".read()" to read the imported file and store it as a string variable.

```
# 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

Next, I converted that file string variable into a list with the ".split()" function.

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

Iterate through the remove list

Here's the defined list of IP addresses to remove, stored in "remove_list".

```
# 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"]
```

I iterated and printed the list with the code below.

```
for element in remove_list:
    # Display `element` in every iteration
    print(element)
```

Remove IP addresses that are on the remove list

I constructed a "for" loop to iterate through the list of IP addresses and remove any IP addresses that were listed in the "remove_list" based on an "if" condition.

```
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)
```

Update the file with the revised list of IP addresses

First, I had to convert the “ip_addresses” variable from a list back into a string so that I could rewrite the original file.

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

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

Next, I rewrote the “import_file” with the updated IP address list.

```
# 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

I created an algorithm that removes IP addresses from an allow list. I opened and read the “allow_list.txt” file that tracks the allowed IP addresses. Next, I converted it into a string and then into a list stored in the “ip_addresses” variable using “.split().

I then iterated through the “remove_list”. If an IP address was in the “remove_list”, I removed it from “ip_addresses” using “.remove()”.

Finally, I converted the “ip_addresses” list back into a string using “.join()”. I used this “ip_addresses” string to update the original “allow_list.txt” file to remove the users that did not have access.