Isaac Aylor
Google Certificate
Python3

Algorithm for file updates in Python

Project description

Within this project, I was tasked with improving our operational security by the removal of previous employee's IP addresses. These addresses were still within our approved list, so I developed a Python algorithm to remove them efficiently.

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"]

# 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)
```

```
ip_address
192.168.25.60
192.168.205.12
192.168.97.225
192.168.6.9
```

Firstly, in order to read the contents of the file `allow_list.txt`, I created a variable named `import_file` and copied the files contents to this variable (line 3). Once provided with the addresses to remove, I created a list containing them called `remove_list` (line 7).

Read the file contents

Utilizing the `with` command (line 11), I specified the `open` command to parse `import_file` with the additional `"r"` argument so to read it into `file`; a temporary function local file. I then put the contents of `file` into a variable named `ip_addresses` using the `.read()` function (line 15). Finally, this enabled me to use the `print(ip_addresses)` function (line 19) to display the contents of `allow_list.txt` in a readable format.

Convert the string into a list

```
ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

```
['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', '192.16
8.90.124', '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']
```

Taking `ip_addresses` and replacing its value with `ip_addresses.split()` (line 1), I transformed its previous string data into a list. By default, the command separated each list item where a space character was. This can be visually seen from `print(ip_addresses)` results (below line 5).

Iterate through the remove list

```
for element in ip_addresses:

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

  if element in remove_list:
```

I then created an iterative `for` function to specify if an `element` (`element` represents each address one by one for the loop) is in the `ip_addresses` (line 1) and the `remove list` (line 6).

Remove IP addresses that are on the remove list

```
  if element in remove_list:

    # then current element should be removed from `ip_addresses`

    ip_addresses.remove(element)
```

If an IP address was in both the `ip_addresses` list and the `remove_list` (line 1), then the loop used `ip_addresses.remove(element)` to remove that address from the `ip_address` list (line 5).

Update the file with the revised list of IP addresses

```
  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", ip_addresses)

# 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)
```

```
192.168.25.60 192.168.6.9 192.168.90.124 192.168.133.188 192.168.218.219 192.168.156.224 192.168.69.116
```

To update the original `allow_list.txt` file with the secure IP address list, I used the `" ".join(ip_addresses)` to transform `ip_addresses` from list back to string data (line 1). This string data was rewritten back to the original variable name (line 7). With my custom defined function `update_file()` (line 13), I updated the contents of `allow_list.txt` with the contents of `ip_addresses`. To verify the results, `open()` with the `"r"` read argument was once more used to copy `allow_list.txt` to a local temporary `text` variable (line 21), allowing to display the updated list with `print()` (line 25)

Summary

This algorithm bolstered operational security, by removing IP access of previous employees. Time was saved by implementing Python rather than manually sifting. Furthermore, developing such code to maintain an accurate address list regularly should be investigated and implemented.