Algorithm for file updates in Python

Project description

In this lab, I will pretend to be a security professional working at a health care company. As part of my job, I will regularly update a file that identifies the employees who can access restricted content. The contents of the file are based on who is working with personal patient records. Employees are restricted access based on their IP address. There is an allowlist for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees I must remove from this allow list.

My task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, I should remove those IP addresses from the file containing the allow list.

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"]

# First line of `with` statement

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

First we assign the variable `import_file` to the name of the file as a string, which is called allow_list.txt, for later use. Later in the code, the allow_list.txt file will contain IP addresses of employees who can access restricted content. Then we assign remove_list to a list of IP addresses that are no longer allowed to access restricted information.

Next, we write the first line of the 'with' statement to open the imported file (which is assigned to the string allow_list.txt) in read mode. The with keyword ensures that the file will open and will close once it is not in use anymore. The second parameter of the open function is 'r' so we can open the file in read mode. The opened file will be stored as a string in the variable file.

Read the file contents

```
# 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)
```

Afterward, I used the .read( ) method to convert the contents of the allow list file into a string so I can read them. I store this string in a variable called ip_addresses. When running the code, nothing will be output because the file is currently emptied. Here is how the code is written: ip_addresses = file.read().

Convert the string into a 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()
# Use `.split()` to convert `ip_addresses` from a string to a list
ip_addresses = ip_addresses.split()
# Display `ip_addresses`
print(ip_addresses)
```

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, I used the .split() method to convert the ip_addresses string into a list,

`ip_addresses = ip_addresses.split()` . When displayed, the output is `[]` because the list is currently emptied.

Iterate through the remove 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()

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

ip_addresses = ip_addresses.split()

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

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

A second list called `remove_list` contains all of the IP addresses that should be removed from the `ip_addresses` list. I set up the header of a for loop that will iterate through the `remove_list`. I use 'element' as the loop variable:

`for element in ip_addresses:`

`  # Display `element` in every iteration`

`  print(element)`

Remove IP addresses that are on the remove 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()

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

ip_addresses = ip_addresses.split()

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

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)

# Display `ip_addresses`

print(ip_addresses)
```

In the body of the iterative statement, I wrote codes that will remove all the IP addresses from the allow list that are also on the remove list. First, I create a conditional that evaluates if the loop variable element is part of the ip_addresses list. Then, within that conditional, I apply the .remove() method to the ip_addresses list and remove the IP addresses identified in the loop variable element:

```
# 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)

# Display `ip_addresses`

print(ip_addresses)
```

Update the file with the revised list of IP addresses

```
# 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()

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

ip_addresses = ip_addresses.split()
```

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

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)

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

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)
```

Now that I have removed these IP addresses from the ip_address variable, I can complete the algorithm by updating the file with this revised list. To do this, I convert the ip_addresses list back into a string using the .join() method. I apply .join() to the string "\n" in order to separate the elements in the file by placing them on a new line.

Then, I use another with statement and the .write() method to write over the file assigned to the import_file variable.

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

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)
```

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. This algorithm involved opening the file, converting it to a string to be read, and then converting this string to a list stored in the variable ip_addresses. I then iterated through the IP addresses in remove_list. With each iteration, I evaluated if the element was part of the ip_addresses list. If it was, I applied the .remove() method to it to remove the element from ip_addresses.. After this, I used the .join() method to convert the ip_addresses back into a string so that I could write over the contents of the "allow_list.txt" file with the revised list of IP addresses.

Writing algorithm requires
- setting up files to be used and preparing logs for analyzing.
- parsing the data
- analyzing it

Python skills demonstrated:

- with `open()` functions
    - `"r,"` `"w"`
- Iterative statements
- conditional statements
- def keyword
- Methods
    - `.read( )`
    - `.write( )`
    - `.split( )`
    - `.join( )`