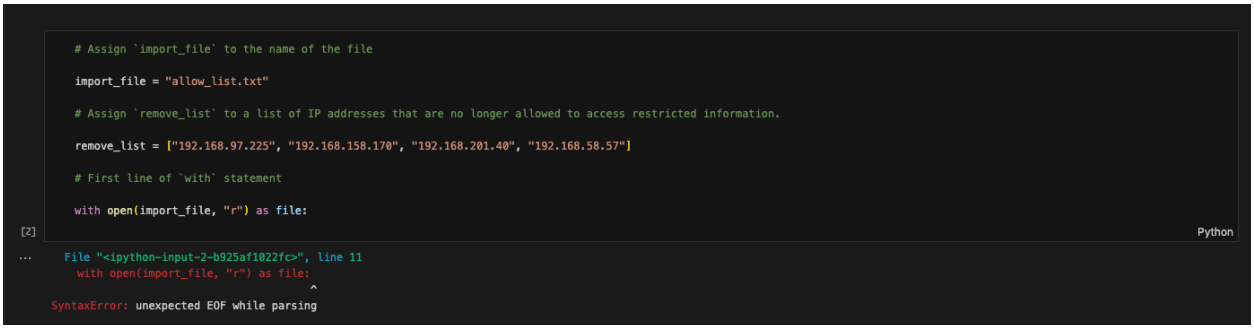
Algorithm for File Updates in Python

Project description:

I am an entry level security professional working at a health care company. I am responsible for maintaining access to the restricted content. Meaning I must update the allowed IP address list and remove IP address list regularly to ensure compliancy and confidentiality. To aid me in this task, I will create an algorithm in Python to automate this repetitive mandatory process.

Open the file that contains the allow list:

The file that you want to open is called `"allow_list.txt"`. Assign a string containing this file name to the `import_file` variable. Then, use a `with` statement to open it. Use the variable `file` to store the file while you work with it inside the `with` statement.



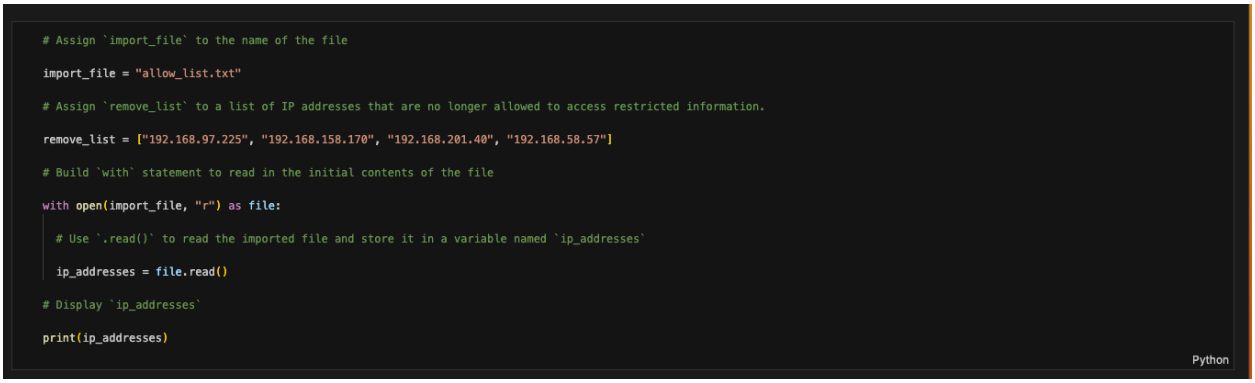
```
# Assign `import_file` to the name of the file

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:
```

[2] Python

```
...   File "<ipython-input-2-b925af1022fc>", line 11
        with open(import_file, "r") as file:
                                            ^
SyntaxError: unexpected EOF while parsing
```

Read the file contents:

Next, use the `.read()` method to convert the contents of the allow list file into a string so that you can read them. Store this string in a variable called `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()

# Display `ip_addresses`

print(ip_addresses)
```

Python

Convert the string into a list:

To remove individual IP addresses from the allow list, the IP addresses need to be in a list format. Therefore, use the `.split()` method to convert the `ip_addresses` string into a list.

```python
# 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)
```

Loading...

Python

Iterate through the remove list:

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

```python
# 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 `remove_list`

for element in remove_list:

  # Display `element` in every iteration
    print(element)
```

Python

Remove IP addresses that are on the remove list:

In the body of your iterative statement, add code that will remove all the IP addresses from the allow list that are also on the remove list. First, create a conditional that evaluates if the loop variable `element` is part of the `ip_addresses` list. Then, within that conditional, apply the `.remove()` method to the `ip_addresses` list and remove the IP addresses identified in the loop variable `element`. In addition, include a sentence that explains that applying the `.remove()` method in this way is possible because there are no duplicates in the `ip_addresses` list.

```python
# 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 `remove_list`

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`

        ### YOUR CODE HERE ###

# Display `ip_addresses`

print(ip_addresses)
```

Python

Update the file with the revised list of IP addresses:

Now that you have removed these IP addresses from the `ip_address` variable, you can complete the algorithm by updating the file with this revised list. To do this, you must first convert the `ip_addresses` list back into a string using the `.join()` method. Apply `.join()` to the string `"\n"` in order to separate the elements in the file by placing them on a new line. Then, use another `with` statement and the `.write()` method to write over the file assigned to the `import_file` variable.

```python
# 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 `remove_list`

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)

# 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)
```

Python

Summary:

I wrote this script / algorithm to review the contents in two variables against each other and to remove those items from the primary list if they were present on the other list. The script starts by opening a file that contains the original list and reads it to a file. The contents of the file are converted into a string which are then converted into a list. The list is then put through a for statement with an if statement. This for loop checks if each item in original list is present in the other list, if is it then that item is removed from the original list. Once the `for` statement has made it through main list, it will then produce a new list. This new list will then overwrite the original file once the list has been converted back into a string.