Algorithm for file updates in Python

Project description

In the lab scenario, I'm a security analyst working at a health care company. It's my responsibility to update a file that identifies employees that are allowed access to restricted information (i.e. patient medical records/PII). The employees are given access based on their IP address. I've been tasked with creating an algorithm that identifies and removes IP addresses from the allow list that should no longer have access to the restricted subnetwork.

Open the file that contains the allow list

I start by assigning values to two variables. I assigned the 'import_file' variable with a value of a text file named 'allow_list.txt' and for the 'remove_list' variable, the value stored is the IP addresses that need to be removed from the allow list.

I then used the keyword 'with()', (which helps with the handling of files by closing the file after you exit the with statement,) the 'open() function (which allows you to open files into Python.) Within 'open()' I passed two arguments, the first is the variable name that has the allow list stored in it, and the second is "r" which tells Python that I want to read the file. I followed this with the 'as' keyword and assigned a temporary variable called 'file' to store the file in.

```
In [2]: # 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:
```

Read the file content

Moving forward, I indented the next line and used the '.read()' method on the temporary variable 'file', to read the file and store it in a new variable called 'ip_addresses'. I then called the 'print()' function and passed 'ip_addresses' in as the argument, resulting in the following output of the "allow_list.txt" file.

While analyzing the file in this structure, I observed that all four of the IP addresses that need to be removed are currently in the allow list.

```
In [3]: # 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
192.168.52.90
192.168.158.170
192.168.90.124
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
```

Convert the string into a list

In order to remove individual IP addresses from the allow list, the IP addresses need to be in a list format. To do this, I exit the ‘with’ statement and reassign the ‘ip_addresses’ variable with the ‘.split()’ method, (This method converts string data types into list data types.) It does this by breaking up the elements of a string based on a specified character that is passed into the method as an argument. Or, if no arguments are passed into the ‘split’ method, every time it encounters a whitespace, it will separate the string.

After this, I called the ‘print()’ function and again passed the ‘ip_addresses’ variable in as the argument.

After running the code you can see that the string elements from before have been converted into a list format in brackets with each element separated and placed in single quotation marks.

In [4]:
```
# 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)
```

```
['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', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.1
68.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

Iterate through the remove list

Here, I'm starting the code to remove the elements of the 'remove_list' variable from the 'IP addresses' variable. To begin this process I'll write an 'iterative statement,' (Iterative statements repeat code for a specified sequence.)

I use the 'for' keyword to tell Python that I'm starting a 'for loop', I then assign the loop variable as 'element' to temporarily store the elements of the list while inside of the 'for loop'. After this, I use the 'in' keyword to tell Python to iterate through the 'ip_addresses' variable.

I then call the 'print()' function with 'element' passed into it as an argument and run the code. As you can see, It outputs the 'element' variable in every iteration, or, in other words, It outputs all the elements from the 'ip_addresses' variable with each element separated on its own line.

```
In [6]: # 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)
```

```
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
192.168.90.124
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
```

Remove IP addresses that are on the remove list

The next step is to create a conditional statement. I start by indenting the next line to tell Python that the conditional statement is happening inside of the ‘for loop’, then I use the ‘if’ keyword to tell Python that we’re starting the conditional statement. After this we use the ‘element’ variable that was previously defined in the ‘for loop’, and the ‘in’ keyword to tell Python to iterate through the ‘remove_list’ and search for elements stored in the ‘element’ variable.

I move to a new line and indent this line further to write the condition inside of the ‘if’ statement. I use the ‘Ip_addresses’ variable with the ‘.remove()’ method, (this method removes an element from a list, because there are no repeating values in the ‘ip_addresses’ variable, this is an instance where you can use the method without passing an argument into it.) This condition tells Python that if any of the elements in the ‘element’ variable are found in the ‘remove_list’, to remove them from the ‘ip_addresses’ variable. Now we have our completed algorithm to remove IP addresses that no longer have access to the restricted information.

After running the code, I then call the ‘print()’ function with the ‘ip_addresses’ variable as the argument to display the contents. As you can see, all four of the IP addresses from the ‘remove_list’ variable are not returned in the output showing that the algorithm worked successfully.

```
In [7]: # 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)
```

```
['ip_address', '192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.52.90', '192.168.90.124', '192.168.
186.176', '192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.1
68.60.153', '192.168.69.116']
```

Update the file with the revised list of IP addresses

The last thing I need to do is convert the ‘ip_addresses’ variable back into a string data type and then overwrite the original allow list ‘allow_list.txt’ with the revised list, (now stored as a string data type) inside of the ‘ip_addresses’ variable.

The first step I take to achieve this task is to reformat the ‘ip_addresses’ variable with the ‘.join()’ method, (which is a method that concatenates the elements in a list into a string data type.) I pass in the ‘.join()’ method following the string “ “ to indicate to leave a space between each element. I pass the ‘ip_addresses’ variable in as the argument in the ‘.join()’ method to tell Python that I want to reformat the variable to a string.

Next, I start another ‘with’ statement to open the original file to write to it. I use the ‘with’ keyword with the ‘open()’ function, and I pass the ‘import_file’ variable, (which contains the original allow list,) in as the first argument, and I pass in ‘w’ to tell Python that I want to write to the file and replace the current contents of the file as the second argument. After, I use the keyword ‘as’ and assign a temporary variable called ‘file’ to store the file while inside of the ‘with’ statement.

I indent the next line and use the temporary variable 'file' with the '.write()' method and pass the 'ip_addresses' variable, (which contains the revised allow list,) in as the argument, with which Python will replace the current contents of the file. After running the code, the revised allow list is now stored in the 'allow_list.txt' file which is stored inside the variable 'import_file'.

In [8]:
```
# 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)
```

Summary

My goal was to write an algorithm that removed a list of IP addresses from a list that allowed access to restricted information. I used 'with' statements to open files and either read them into another variable for later use with 'r' as an argument, or, overwrote the contents of them with 'w' as an argument. I also applied both the '.read()' and '.write()' methods to help complete this step of the process, as well as the '.split()' method to parse and reformat the elements of the allow list

I then combined iterative and conditional statements as the next step to writing the algorithm. Using a for loop that contained an if statement with a condition to remove elements of the 'allow_list' variable if they matched the elements stored in the 'remove_list' variable.

After running the code containing the algorithm, I converted the list back into a string with the '.join()' method and stored the string in a variable, which I then used it to overwrite the original allow list with the revised version.