Update a file through a Python algorithm

In this project, I will be explaining a step-by-step process about how I was able to open a file, read the contents within the file, write contents within the file, and create an algorithm that can update a file using Python.

Scenario

I am a security professional working at a health care company. My job is to 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 allow list of 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 was to develop a Python algorithm to remove IPs listed on the remove list from the "allow_list.txt".

Step 1: Open the file that contains the allow list

For the first part of the algorithm, I opened the "allow_list.txt" file. First, I assigned this file name as a string to the import_file variable:

```
# Create a variable 'import file' and assign it to the name of the file.
import_file = "/users/tetteh/allow_list.txt"

# 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"]

```

Then, I used a with statement to open the file:

```
# Build 'with' statement to read in the initial contents of the file
with open(import_file, "r") as file:

```

The with statement is used with the .open() function in read mode to open the allow_list.txt file for the purpose of reading the contents inside. In the code with open(import_file, "r") as file:, the open() function has two parameters. The first identifies the file to import, and then the second indicates what I want to do with the file. In this case, "r" indicates that I want to read it. The code also uses the as keyword to assign a variable named file; file stores the output of the .open() function while I work within the with statement.

Step 2: Read the file contents

In order to read the file contents, I used the .read() method to convert it into a string.

```
# Create a variable 'import file' and assign it to the name of the file.
import_file = "/users/tetteh/allow_list.txt"

# 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()' function to read the imported file and store it in a variable named 'ip_addresses'
    ip_addresses = file.read()

print(ip_addresses)

```

When using an .open() function that includes the argument "r" for “read,” I can call the .read() function in the body of the with statement. The .read() method converts the file into a string and allows me to read it. I applied the .read() method to the file variable identified in the with statement. Then, I assigned the string output of this method to the variable ip_addresses.

In summary, this code reads the contents of the "allow_list.txt" file into a string format that allows me to later use the string to organize and extract data in my Python program. Below is the output when I read the file. Below is the output.

```
192.168.25.60
192.168.205.12
192.168.6.9
192.168.158.170
192.168.52.90
192.168.90.124
192.168.186.176
192.168.97.225
192.168.133.188
192.168.203.198
192.168.218.219
192.168.52.37
192.168.201.40
192.168.156.224
192.168.60.153
192.168.69.116
192.168.58.57
```

Step 3: Convert the string into a list

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

```
# Create a variable 'import file' and assign it to the name of the file.
import_file = "/users/tetteh/allow_list.txt"

# 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()' function to read the imported file and store it in a variable named 'ip_addresses'
    ip_addresses = file.read()

print(ip_addresses)

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

# Display 'ip_addresses'
print(ip_addresses)
```

The .split() function is called by appending it to a string variable. It works by converting the contents of a string to a list. The purpose of splitting ip_addresses into a list is to make it easier to remove IP addresses from the allow list. By default, the .split() function splits the text by whitespace into list elements. In this algorithm, the .split() function takes the data stored in the variable ip_addresses, which is a string of IP addresses that are each separated by a whitespace, and it converts this string into a list of IP addresses. To store this list, I reassigned it back to the variable ip_addresses. Below is what the output will be like when I convert a string into a list:

```
['192.168.25.60', '192.168.205.12', '192.168.6.9', '192.168.158.170',
'192.168.52.90', '192.168.90.124', '192.168.186.176', '192.168.97.225',
'192.168.133.188', '192.168.203.198', '192.168.218.219', '192.168.52.37',
'192.168.201.40', '192.168.156.224', '192.168.60.153', '192.168.69.116',
'192.168.58.57']
```

Step 4: iterate through the remove list

The next step involves iterating through the IP addresses that are elements in the remove_list. To do this, I incorporated a for loop:

```
# Build iterative statement
# Name loop variable 'element'
# Loop through 'remove_list'

for element in remove_list:
```

The for loop in Python repeats code for a specified sequence. The overall purpose of the for loop in a Python algorithm like this is to apply specific code statements to all elements in a sequence. The keyword element will iterate through each of the ip_addresses in the remove_list.

Step 5: Remove IP addresses that are on the remove list

My algorithm requires removing any IP address from the allow list, ip_addresses, that is also contained in remove_list. In this step, I will create an if statement inside the for loop of the remove_list, which will check and see if the remove list has any matching pairs with the allow_list. If it does then it will remove those IP addresses with the .remove() method. This will check if there are any ip_addreses from the allow list that match with the remove_list. If it does match, I will use the .remove() method which will remove any duplicate IP addresses.

```
for element in remove_list:
    # Create a 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)

# Display 'ip_addresses'
print(ip_addresses)

```

Below is what the output will look like:

```
['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', '192.168.60.153',
'192.168.69.116']
```

First, within my for loop, I created a conditional that evaluated whether or not the loop variable element was found in the ip_addresses list. I did this because applying .remove() to elements that were not found in ip_addresses would result in an error.

Then, within that conditional, I applied .remove() to ip_addresses. I passed in the loop variable element as the argument so that each IP address that was in the remove_list would be removed from ip_addresses.

Step 6: Update the file with the revised list of IP addresses

As a final step in my algorithm, I needed to update the allow list file with the revised list of IP addresses. To do so, I had to convert the list back into a string. I used the .join() method for this:

```
# Convert 'ip_addresses' back to a string so that it can be written to a file
ip_addresses = "\n".join(ip_addresses)

```

Now it should return back into a column that lists each IP address as before, but revised.

```
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
192.168.60.153
192.168.69.116
```

The .join() method combines all items in an iterable into a string. The .join() method is applied to a string containing characters that will separate the elements in the iterable once joined into a string. In this algorithm, I used the .join() method to create a string from the list ip_addresses so that I could pass it in as an argument to the .write() method when writing to the file "allow_list.txt". I used the string ("\n") as the separator to instruct Python to place each element on a new line.

Then, I used another with statement and the .write() method to update the file:

```
38
39  # Build 'with' statement to rewrite the original file
40  with open(import_file, "w") as file:
41
42      # Rewrite the file, replacing its contents with 'ip_addresses'
43      file.write(ip_addresses)
44
```

This will make the file allow_list.txt into a newly reformed list of IP addresses that I have just overwritten with Python.

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.