Algorithm for file updates in Python

Project description

As a security professional at a healthcare company, my task is to create a Python algorithm to update the file that manages employees' access to restricted content. The file contains an allow list of permitted IP addresses and a remove list of employees to be removed from the allow list. To achieve this, I'll use Python code to check if any IP addresses from the remove list are present in the allow list and remove them if necessary. This ensures that only authorized employees can access personal patient records on the restricted subnetwork.

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:
```

```
  File "<ipython-input-6-b925af1022fc>", line 11
    with open(import_file, "r") as file:
                                        ^
SyntaxError: unexpected EOF while parsing
```

1.First I assigned the file “allow_list.txt” to the variable “import_file” as you can see above.
2.Then I created a list “remove_list” for all the IP addresses that are no longer allowed access to restricted information
3. I used a “with” statement combined with a “open” function to open the “import_file”. The end of the code I used “r” which is the ‘read’ mode and then finally “as file” to create a file object named ‘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)
```

1.Still within the ‘with’ block ‘file.read’ is used to read the entire content of the file whilst also storing it in a variable named “ip_addresses”
2. This prints the data of ip_addresses and displays all the content on the screen for the user to see

Convert the string into a list

```
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', '192.168.15
8.170', '192.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.2
19', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

1.After reading the file and storing the data to the variable ‘ip_addresses” the following code uses the .split method which converts the list of ip addresses into individual lines of ip addresses. The split method splits the string by whitespace and then creates a new line for that content.

Iterate through the remove list

```
# 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)
```

1.From the top a remove_list has already been created
2. Here we used a 'for loop', the loop variable is named element and it is assisgned to each individual ip address
3.The last code "print(element)" prints or displays the loop variable element which shows each individual ip address that is in the "remove_list" string.

Remove IP addresses that are on the remove list

```
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)

# Display `ip_addresses`

print(ip_addresses)
```

1. After creating the code line “for element in remove_list:” I then used a “if” statement
2. The “if” statement checks if the ip addresses in the remove_list are also in the ip_addresses list.
3. If the statement above is correct, then the following code is applied “ip_addresses.remove(element)”. This essentially removes one by one the ip addresses that need to be removed from the original list.
4. Last code simply displays the updated list of ip addresses after the removal of ips from “remove_list”

Update the file with the revised list of 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)

# Build `with` statement to read in the updated file

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

    # Read in the updated file and store the contents in `text`

    text = file.read()

# Display the contents of `text`

print(text)
```

1. After removing all the ip addresses the code “ip_addresses = "\n".join(ip_addresses)” converts the list back into a string using the .join method.
2. Then the ‘with’ statement opens the original file of ip addresses and rewrites the file. The “w” in the code allows the write method and therefore the file can be modified.

Summary

In my role as a cybersecurity expert at a healthcare company, I have been assigned the task of developing a Python algorithm to keep the employees' access to sensitive content up-to-date. This file consists of two lists: an "allow list" containing approved IP addresses and a "remove list" specifying employees to be taken off the allow list. To accomplish this, I utilized Python code to cross-reference the remove list with the allow list and remove any matching IP

addresses. By doing so, we can ensure that only authorized personnel can access personal patient records on the restricted network.