Algorithm for file updates in Python

Project description

So, this is just a project to simulate day-to-day work as a security analyst. We'll work with python a bit. For this project, we'll create a functional algorithm that parses a text file of IP addresses. The algorithm should also update the file by removing addresses that no longer have access to the restricted content.

Open the file that contains the allow list and read file contents

Here we're assigning a string containing the "allow_list.txt" file to the 'import_file' variable. We also use a with statement for good coding practice, and open to you know, open the file. Then we'll use the file object we created so that we can use the .read function and store the text in a string variable called 'ip_addresses'. Remember, .read() doesn't change any property in the file.

```
# 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)
```

Convert the string into a list

In this part we'd like to individualize the IP addresses so that we may remove them. Strings are immutable, but lists aren't. We'll use the .split() method to convert the 'ip_addresses' 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()

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

for element in ip_addresses:

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

Iterate through the remove list

We have a second list called 'remove_list' that contains the IP addresses that should be removed from the original 'ip_addresses' list. Our job in this step is to set up a for loop so that we can iterate through the 'ip_addresses' list. Notice how we use the object named 'element' as our iterator to move throughout the 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)
```

```
ip_address
```

Remove IP addresses that are on the remove list

Now we'll remove the necessary IP addresses from the newly created IP address list. To do this, we prepped the for loop to iterate through 'ip_addresses' list. Now, we've added a conditional if statement, testing if the current element that we're looking at matches the same element found in the remove list. If the element is found in the remove_list, then we'll use the '.remove()' function with the element added into the function as the one argument. '.remove()' is used on the 'ip_addresses' variable, and the one argument that the function should accept should be the list element that matches with the element found in the 'remove_list'.

```
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)
```

Update the file with the revised list of IP addresses

After taking care of removing the necessary IP addresses, we'll need to update the ip_addresses list. We'll have to convert the list back into a string. Remember, the .join() method is the iterable we'd like to convert, in this case – it's ip_addresses. After the string conversion, we'll write back into the 'import_file' variable, which contains the 'allow_list.txt' information about the ip addresses. We'll write, so the second argument used in the open() method should be "w". In this case, writing into the text file will replace all the original text with the revised version.

```
  # 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

So, now I have practical experience with file operations in python. I've mostly used python for numerical methods and engineering math, so to finally understand python in a cyber security analyst setting is like filling in a missing gap to my knowledge. Now, I understand the types of algorithms I'll need to know to file updates.