ALGORITHM FOR FILE UPDATES IN PYTHON

PROJECT DESCRIPTION

You are a security professional working at a health care company. As part of your job, you're required 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 for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list.

Your task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, you should remove those IP addresses from the file containing the allow list.

OPEN THE FILE THAT CONTAINS THE ALLOW LIST

In Python it is possible to open a file and add or remove information, or just check the contents. For the last action, using the `.read()` method.

In the example below, the file name is stored in the variable. The IP address is not in the file and needs to be removed stored in the `remove_list` variable.

After that, the `with` statement line is the header and contains the `open` statement to open a file. The first argument is the file name - in this case the name is stored in the `import_file` variable. The second argument determines what to do with the file - in this case, the letter `“r”` reads a file. The `as` in the sequence with `file` is the argument syntax.

```
# 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 CONTENTS

In the next, it's create a variable `ip_addresses` that stores the results of the `.read()` method. Using `print`, the output displays the result in the next screenshot.

```
# 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

For converting the `ip_address` in a list, the `.split()` method it's used. This method converts each line into a list separated by commas. Note the example in below:

```
# 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', '1
92.168.90.124', '192.168.186.176', '192.168.133.188', '192.168.203.198', '192.168.201.40', '192.168.218.219', '192.168.5
2.37', '192.168.156.224', '192.168.60.153', '192.168.58.57', '192.168.69.116']
```

ITERATE THROUGH THE REMOVE LIST

To check each element in the `ip_address`, a `for` loop is used to iterate and go through each element and display on the screen.

```
# 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)
```

REMOVE IP ADDRESSES THAT ARE ON THE REMOVE LIST

To remove IP addresses, the `.remove()` method is applied. In this case, the `remove_list` variable stores IP addresses that cannot be in the `ip_address` variable. In the for loop, the `if` statement checks whether the elements match the `ip_address`. If true, the method is used and passes an argument. **OBS**: this method it's useful if it does not contain repetition of elements.

```
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', '192.168.1
33.188', '192.168.203.198', '192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

UPDATE THE FILE WITH THE REVISED LIST OF IP ADDRESSES

In the last step, an update is required to save all changes. For this, the `.join()` method is used to convert `ip_addresses` back into a string so that it can be written to the text file. Next, it is necessary to create another `with` statement with arguments for writing and another for reading. Note this below. The result of this operation is on the right.

```
# 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)
```

```
ip_address
192.168.205.12
192.168.6.9
192.168.52.90
192.168.90.124
192.168.186.176
192.168.133.188
192.168.218.219
192.168.52.37
192.168.156.224
192.168.60.153
192.168.69.116
```

SUMMARY

This project presented how the Python language automated repetitive tasks. In this case, using methods such as `.read()`, `.write()`, `.join()` and `.split()` and using the `with` statement, in addition to the `for` loop and the `if` statement, allowed improvements in data analysis. Furthermore, to remove any information, the `.removed()` method was also used.