Algorithm for file updates in Python

Project description

My job was to regularly update the files by checking employees who should have the right access to restricted content. I used python to create automations of removing any ip addresses that match one of the list of ip addresses that needs to be removed. I accomplished using various keywords/functions to parse the file of ip addresses in the way that is easily manipulated.

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:
```

The following screenshot shows an attempt to open a file that contains the allow list. The `#` symbol represents a comment that tells the purpose of the code. The code `import_file = "allow_list.txt"` represents assigning the file name "allow_list.txt" to the variable named import_file. There's a list assigned to a variable named removed_list, but that is not necessarily relevant in this section. Finally, the code `wide open(import_file, "r") as file:` represents attempting to open and read the file "allow_list.txt" and assign the name as file. Let's break it down:

- **with** - keyword that handles errors and manages external resources when used with other functions like open()
- **open(import_file, "r")** - open() is a function that takes in two parameters: The name of file or variable name of the file and a letter "r" indicates that I want to read the file
- **as file** - is to provide a variable so that I can store within the "with" statement.

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)
```

In this screen shot, I have assigned a variable to read the file and print it to display the file contents. The code `ip_address = file.read()` represents that I used 'file.read()' to read the variable named "file" from the "with" statement and assign it to a variable named 'ip_addresses'. The code `print(ip_addresses)` displays the content of the file from 'import_file' in order to view the information inside the file.

Convert the 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()
# Display `ip_addresses`
print(ip_addresses)
```

In order to convert the string into a list, I have to apply the `.split()` keyword. What the keyword `.split()` does is it converts the string into a list. What the code `ip_addresses = ip_addresses.split()` does is it takes the variable that we made previously and separates individual string of text into a list that is separated by a whitespace or per line since there is no argument in the `.split()` keyword and assigned it to the variable "ip_addresses". We then use the code `print(ip_addresses)` to display the list of ip addresses.

Iterate through the remove 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)
```

In this screenshot, I want to iterate through the remove list. Before that, I will explain what the code `for element in ip_addresses:` does. The **for** keyword is the start of a loop function that iterates through a definite amount of time within the for loop statement. The **element** is a loop variable that is used to qualify each individual element to pass on or print. The ip_addresses goes through each element in the list of ip_addresses that we have created from before. Within the for loop function, we displayed each element in every iteration of the list ip_addresses.

Remove IP addresses that are on the remove 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:

  # 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)
```

The screenshot above shows an attempt to remove an element from “ip_addresses” for every individual element that matches in the “remove_list”. The code `if element in remove_list”:` is a conditional statement that performs the code within the if statement if the element matches to one of the elements in remove_list. Followed by `ip_addresses.remove(element)`, which removes the element under ip_addresses list.

The purpose of `.remove(element)` is to remove an individual element (IP address) from the ip_addresses list if an element that is being iterated matches one of the elements in the removed_list variable.

Update the file with the revised list of IP addresses

```
# 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)
```

After checking for any IP addresses that I need to remove, I must convert the list ip_addresses back to a string. To do this, I use the following code `ip_addresses = " ". join(ip_addresses)`. In order to convert the list back to string, I must use the .join()

keyword to convert the list into a string using ip_addresses as an argument and followed by a white space “ ” to indicate to be converted each element separated by a space.

After converting back into a string, I would need to update the original file by using the following code: `with open(import_file, “w”) as file:`. In this code, I would use the with statement to open the import_file with “w” to write over the file (just like replacing the content) as file.

Within the “with” statement, I would have to update the file using the code `file.write(ip_addresses)`. The `.write()` keyword is used to update/write over the file that is passed in from ip_addresses to the name file so that it replaces the content with ip_addresses.

Summary

I would like to give a quick recap of the components used to develop this algorithm. Firstly, I opened the `import_file` to be read to store in the variable `ip_addresses`. Secondly, I convert the string into a list to start the iteration process. Thirdly, I created a for loop to check if each individual element in the ip addresses match in the remove list and remove any elements that are satisfied through the if statement. Lastly, I have converted back into a string and updated the file to its current condition.