Renaldo Schmidt

Algorithm for File Updates in Python

Project description

In this project, I aim to demonstrate my ability to automate tasks using Python to employers. In this scenario, I am a security professional in a healthcare company, where part of my job is to regularly update a file that identifies employees who can access restricted content. The employees gain access to the content based on their IP address, and there is a list of IP addresses that need to be removed from the approved list file. I will use Python to create an algorithm that removes the IP addresses of those who no longer need access to the restricted content from the approved IP addresses file.

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:
```

In the first line of code, I created a variable named “import_file” and assigned it to the file “allow_list.txt”.

The second line of code assigns a list of IP addresses that need to be removed to a variable called “remove_list”. Lists are contained within brackets and the elements are separated by commas.

The third line of code is the header of a function where it will read the “allow_list.txt” file. It starts with the “with” keyword, which is important to use because it will close the file after exiting the “with” statement. Then the “open” function comes after and contains two parameters contained within parenthesis. The first parameter is the file in which to read or modify, and in this case, the variable “import_file” is passed in which contains the file “allow_list.txt”. The second parameter defines what you want to do to the file, and in this case, it is “r”, which states that we will read the file’s contents. After the “open” function comes the “as” keyword, which will store the file in a variable specified following it while working within the “with” statement. Here the variable that will temporarily store the file is named “file”.

Read the file contents

```
# 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)
```

Now we will create the body of the “open” function so we can read the contents of the file. The body contains a variable called “ip_addresses” which is assigned to the “read” method of the file. This stores the contents of the file as a string in the “ip_addresses” variable.

```
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
```

This is the result of printing the variable “ip_addresses”, which contains the contents of the file we just read as a string.

Convert the string into a list

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

ip_addresses = ip_addresses.split()
```

In order to convert the string data of the file we read into a list, we can reassign the variable we just stored it in.
This is done by assigning the “ip_addresses” variable to itself and adding a “split()” method at the end. There is no argument within the parenthesis of the split method, so it will separate the elements of the list based on white spaces in the original string.

```
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', '192.16
8.90.124', '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']
```

Now if we print the variable “ip_addresses”, it contains a list of the data from the “allow_list.txt” file.

Iterate through the IP addresses list

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

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

Now we will iterate through the list by using a “for” loop.
After the “for” function comes the loop variable that we are going to use within this statement, which in this case is called “element”.
Then comes the “in” keyword, which is used to specify which list we are going to iterate through.
After the “in” keyword comes the list, which in this case is stored in the variable “ip_addresses” that we created and converted into a list in earlier steps.

```
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
```

After printing the loop variable “element” which is contained in the iterative statement, it returns each element of the list one by one in its own line.

Remove IP addresses that are on the remove list

```
# 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)
```

Now we will create a conditional statement using the “if” function within the iterative, in order to remove the IP addresses that are also included in the remove list.
The loop variable “element” is placed after “if” to specify that we are comparing the elements within the “ip_addresses” list to the elements “in” the “remove_list”.
Then we create the body of the conditional statement to tell the program what to do when elements of both lists match. In this case we use the “.remove()” method after the “ip_addresses” variable to remove the “element”s that are contained within both lists.

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)
```

Now we will convert the contents of the “ip_addresses” variable back into a string so that we can update the contents of the original “allow_list.txt” file. This is important because you can not write list data into a file.
To convert it from a list to a string, the “.join()” method is used. This is done by assigning the “ip_addresses” variable to “” “.join(ip_addresses)”.
The join method is different from the split method because you put the file in which you are converting in the parenthesis, and it is added to the end of the character with which you are going to separate each element with once it is joined. The separation character within the

quotation marks is a space, so each element will be separated by a space when it is converted into a string.
Now we can update the contents of the “allow_list.txt” file by using a “with open” function and putting the variable that we have been modifying “ip_addresses” as the first parameter, and “w” as the second to tell the program to write to the file. Then the file is temporarily saved as the variable “file” by using “as”.
Then the body of the function tells the program to write to the file using the contents of the “ip_addresses” variable. The variable is passed in as the argument within the parenthesis of the “.write()” method, and the local variable “file” is placed before the “.write()”.

Summary

In this project, I created an algorithm that updates a file containing a list of IP addresses of allowed users using Python. The main components of this algorithm are the “.read()” and “.write()” methods which allowed me to get and update the data from the file after using the “open” function. More major components are the “if” and “for” conditional and iterative statements which allowed me to remove unwanted IP addresses by comparing elements of different lists. Also the “.split()” and “.join()” methods were useful in converting the data type of the IP addresses so that I could work with them using the conditional and iterative statements. Lastly, the “.remove()” method was used in the iterative and conditional to remove the unwanted IPs. To go a step further I could define a function named “update_file” that runs this whole algorithm based on two parameters, in order to streamline the whole process, as shown below.

```python
# Define a function named `update_file` that takes in two parameters: `import_file` and `remove_list`
# and combines the steps you've written in this lab leading up to this

def update_file(import_file, remove_list):

  # 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)

  # 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)
```