Algorithm for file updates in Python

Project description

Work with a text file containing IP addresses that are allowed to access specific restricted content within a potential organization.

Parsing a file allows security analysts to read and update the contents. Python helps analysts develop algorithms to automate the process of parsing files and keeping them up-to-date.

Develop an algorithm that parses this text file of IP addresses and updates the file by removing addresses that no longer have access to the restricted content.

Open the file that contains the allow list

start by opening the text file using the `import_file` variable, the `with` keyword, and the `open()` function with the `"r"` parameter.

Write the first line of the `with` statement. Running this code will produce an error because it will only contain the first line of the `with` statement.

```
# 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(allow_list.txt, "r") as file:
```

The `open()` function in Python allows you to open a file.

As the first parameter, it takes in the name of the file (or a variable containing the name of the file).

As the second parameter, it takes in a string that indicates how the file should be handled.

Passin the letter `"r"` as the second parameter will make the file available to be read.

Read the file contents

Use the `.read()` method to read the imported file and store it in a variable named `ip_addresses`.

Afterwards, display `ip_addresses` to examine the data in its current format.

```
# 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
```

The `.read()` method in Python allows you to read in a file.

Call `file.read()` to read the imported file.

To display the contents of a variable, pass it as an argument to the `print()` function.

Convert the string into a list

After reading the file, reassign the ip_addresses variable so its data type is updated from a string to a list. Using the .split() method to achieve this.
Adding this step will allow it to iterate through each of the IP addresses in the allow list instead of navigating a large string that contains all the addresses merged together.
Afterwards, display the ip_addresses variable to verify that the update took place.

```
# 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', '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']
```

The `.split()` method in Python allows you to convert a string to a list. This method can take in a parameter that specifies which character to split on. If a parameter is not passed in, the method will split on whitespace by default.

The default behavior of `.split()` works well. Each IP address is on a new line in the `allow_list.txt` file. When `.split()` is used, it will separate the IP addresses and output them as a list.

To display the contents of a variable, pass it as an argument to the `print()` function.

Iterate through the remove list

Write code that removes the elements of `remove_list` from the `ip_addresses` list. This will require both an iterative statement and a conditional statement.

First, build the iterative statement. Name the loop variable `element`, loop through `ip_addresses`, and display each element.

```
# 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
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
```

Build a `for` loop to iterate through `ip_addresses`. Start with the `for` keyword. Use `element` as the loop variable and use `in` as the loop condition.

To display the contents of a variable, pass it as an argument to the `print()` function.

Remove IP addresses that are on the remove list

Build a conditional statement to remove the elements of `remove_list` from the `ip_addresses` list. The conditional statement should be placed inside the iterative statement that loops through `ip_addresses`. In every iteration, if the current element in the `ip_addresses` list is in the `remove_list`, the `remove()` method should be used to remove that element.

Afterwards, display the updated `ip_addresses` list to verify that the elements of `remove_list` are no longer in the `ip_addresses`.

```
# 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)
```

```
['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', '192.168.133.188', '192.168.203.198',
'192.168.218.219', '192.168.52.37', '192.168.156.224', '192.168.60.153', '192.168.69.116']
```

When building the conditional statement, use the in operator to check if element is in remove_list.

To remove element from ip_addresses, call the .remove() method on ip_addresses, and pass in element.

To remove element from ip_addresses, call ip_addresses.remove() and pass in element.

Update the file with the revised list of IP addresses

The next step is to update the original file that was used to create the `ip_addresses` list. A line of code containing the `.join()` method has been added to the code so that the file can be updated. This is necessary because `ip_addresses` must be in string format when used inside the `with` statement to rewrite the file.

The `.join()` method takes in an iterable (such as a list) and concatenates every element of it into a string. The `.join()` method is applied to a string consisting of the character that will be used to separate every element in the iterable once it's converted into a string. In the code below, the method is applied to the string `" "`, which contains just a space character. The argument of the `.join()` method is the iterable you want to convert, and in this case, that's `ip_addresses`. As a result, it converts `ip_addresses` from a list back into a string with a space between each element and the next.

After this line with the `.join()` method, build the `with` statement that rewrites the original file. Use the `"w"` parameter when calling the `open()` function to delete the contents in the original file and replace it with what you want to write.

```
# 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:
```

```
  # 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)
```

Complete the first line of the `with` statement, call the `open()` function and pass in the name of the file as the first parameter and the letter `"w"` as the second parameter.

The `"w"` parameter specifies that you're opening the file for the purpose of writing to it.

Inside the `with` statement, call the `.write()` method to replace the contents of the file with the data stored in `ip_addresses`.

Inside the `with` statement, call `file.write()` and pass in `ip_addresses`.

Summary

The provided Python code defines a function **`update_file`** that updates the contents of a text file containing IP addresses. The function reads the initial contents of the file, converts them into a list of IP addresses, removes the IP addresses specified in the **`remove_list`**, converts the list back to a string, and rewrites the file with the updated contents. The code demonstrates the use of **`with`** statements and the **`open()`** function to handle file operations safely and efficiently. It also utilizes the **`.read()`** and **`.write()`** methods to read from and write to files, respectively. Additionally, the **`.split()`** method is used to convert the string of IP addresses

into a list, and a `for` loop combined with the `.remove()` method is used to remove specific IP addresses from the list.

    - The `with` statement allows you to efficiently handle files.
    - The `open()` function allows you to import or open a file. It takes in the name of the file as the first parameter and a string that indicates the purpose of opening the file as the second parameter.
        - Specify `"r"` as the second parameter if you're opening the file for reading purposes.
        - Specify `"w"` as the second parameter if you're opening the file for writing purposes.
    - The `.read()` method allows you to read in a file.
    - The `.write()` method allows you to append or write to a file.
- You can use a `for` loop to iterate over a list.
- You can use an `if` statement to check if a given value is in a list and execute a specific action if so.
- You can use the `.split()` method to convert a string to a list.
- You can use Python to compare the contents of a text file against elements of a list.
- Algorithms can be incorporated into functions. When defining a function, you must specify the parameters it takes in and the actions it should execute.