Algorithm for File Updates in Python

Project Description:

In this scenario (as a security analyst), I am responsible for developing an algorithm that parses a file containing IP addresses that are allowed to access restricted content and remove IP addresses that no longer have access. Below is the step-by-step process showing screenshots and a brief description for each step in building the algorithm:

Open the file that contains the allow list:

```
In [ ]: # 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:
```

- First, I began the algorithm by opening the variable `import_file` that contains the "allow_list.txt" file using the `open()` function combined with the `with` statement and a second argument of "r" which indicates to read the file contents.

Read the file contents:

```
In [ ]: # 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)
```

- Then, to read the file contents, I used the `.read()` method to convert it into a string. I applied the `.read()` method to the `file` variable identified in the `with` statement. Then, I assigned the string output of this method to the variable `ip_addresses`.

Convert the string into a list:

Algorithm for File Updates in Python

```
In [ ]: # 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)
```

- To convert the string into a list, I used the .split() function and appended it to a string variable. The purpose of splitting ip_addresses into the list is to make it easier to remove IP addresses from the allow list.
- The .split() function takes the data stored in the variable ip_addresses, which is a string of IP addresses that are each separated by a whitespace, and it converts this string into a list of IP addresses. To store this list, I reassigned it back to the variable ip_addresses.

Iterate through the remove list:

```
In [ ]: # 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)
```

- The purpose of the for loop in the algorithm is to apply specific code statements to all elements in the sequence. It's followed by the loop variable element and the keyword in.

The keyword in indicates to iterate through the sequence ip_addresses and assign each value to the loop variable element.

Remove IP addresses that are on the remove list:

```
In [ ]: # 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)
```

- First, within my for loop, I created a conditional if statement that evaluated whether or not the loop variable element was found in the ip_addresses list. I did this because applying .remove() to elements that were not found in ip_addresses would result in an error.
- Then, within that conditional, I applied .remove() to ip_addresses. I passed in the loop variable element as the argument so that each IP address that was in the remove_list would be removed from ip_addresses.

Update the file with the revised list of IP addresses:

Algorithm for File Updates in Python

```
In [ ]: # 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)

        # 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)
```

- For the final step, I needed to update the allow list file with the revised list of IP addresses. I did this by converting the list back into a string using the .join() method. I used the .join() method to create a string from the list ip_addresses so that I could pass it in as an argument to the .write() method when writing to the file "allow_list.txt".
- Then, I used another with statement and the .write() method to update the file. I used a second argument of "w" with the open() function in my with statement. When using this argument "w", I can call the .write() function in the body of the with statement. The .write() function writes string data to a specified file and replaces any existing file content.
- Finally, I wanted to write the updated allow list as a string to the file "allow_list.txt". This way, the restricted content will no longer be accessible to any IP addresses that were removed from the allow list. To rewrite the file, I appended the .write() function to the file object file that I identified in the with statement. I passed in the ip_addresses variable as the argument to specify that the contents of the file specified in the with statement should be replaced with the data in this variable.

Algorithm for File Updates in Python

Summary:

- I created an algorithm that removes IP addresses identified in a `remove_list` variable from the `"allow_list.txt"` file of approved IP addresses.
- This algorithm involved
    - opening the file
    - converting it to a string to be read
    - and then converting this string to a list stored in the variable `ip_addresses`.
- I then iterated through the IP addresses in `remove_list`.
- With each iteration, I evaluated if the element was part of the `ip_addresses` list.
    - If it was, I applied the `.remove()` method to it to remove the element from `ip_addresses`.
- After this, I used the `.join()` method to convert the `ip_addresses` back into a string so that I could write over the contents of the `"allow_list.txt"` file with the revised list of IP addresses.

Key takeaways from this project include:

- Python has functions and syntax that help you import and parse text files.
    - 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 the `.join()` method to convert lists back into a string.
- You can use Python to compare 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.