Algorithm for file updates in Python

Project Description

My organization wants me to create an algorithm to automate updating a file "allow_list.txt" by removing IP addresses that should no longer have access to the restricted content in this file.

Open the File that Contains the Allow List

The first step is to open the text file containing the IP addresses that are allowed to access restricted information. I used the `open()` function with the "r" parameter to open the file in read mode.

```
import_file = "allow_list.txt"
remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

with open(import_file, "r") as file:
```

Read the File Contents

After opening the file, I used the `.read()` method to read the entire contents of the file into a variable called `ip_addresses`. This allows me to work with the data stored in the file.

```
with open(import_file, "r") as file:
    ip_addresses = file.read()
```

Convert the String into a List

The `.read()` method returns the file contents as a single string. To work with individual IP addresses, I used the `.split()` method to convert the string into a list, where each IP address is a separate element.

```
ip_addresses = ip_addresses.split()
```

Iterate Through the Remove List

To identify and remove IP addresses that should no longer have access, I created a loop that iterates through each element in the `ip_addresses` list.

```
for element in ip_addresses:
```

Remove IP Addresses that are on the Remove List

Inside the loop, I added a conditional statement to check if the current element is in the `remove_list`. If it is, I used the `.remove()` method to remove that IP address from the `ip_addresses` list.

```
for element in ip_addresses:
    if element in remove_list:
        ip_addresses.remove(element)
```

Update the File with the Revised List of IP Addresses

After removing the unauthorized IP addresses, I needed to update the original file. First, I converted the `ip_addresses` list back to a string using the `.join()` method. Then, I opened the file in write mode ("w") and used the `.write()` method to overwrite the file with the updated list.

```
ip_addresses = " ".join(ip_addresses)
```

```
with open(import_file, "w") as file:
    file.write(ip_addresses)
```

Summary

I created a function called `update_file()` that combines all these steps to automate the process of removing unauthorized IP addresses from the allow list file. The function takes two parameters: the name of the file to update and a list of IP addresses to remove.

```python
def update_file(import_file, remove_list):
    with open(import_file, "r") as file:
        ip_addresses = file.read()

    ip_addresses = ip_addresses.split()

    for element in ip_addresses:
        if element in remove_list:
            ip_addresses.remove(element)

    ip_addresses = " ".join(ip_addresses)

    with open(import_file, "w") as file:
        file.write(ip_addresses)
```

This function can be called with different file names and different lists of IP addresses to remove, making it a flexible and reusable solution for maintaining access control lists. By automating this process, I've helped improve the security of our organization's restricted content while saving time and reducing the risk of human error.

THE FULL JUPYTER PYTHON DOCUMENT IS ATTACHED BELOW

Activity_Create another algorithm

March 15, 2025

1 Activity: Create another algorithm

1.1 Introduction

An important part of cybersecurity is controlling access to restricted content. In this lab, you'll work with a text file containing IP addresses that are allowed to access specific restricted content at your 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.

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

Tips for completing this lab

As you navigate this lab, keep the following tips in mind:

- `### YOUR CODE HERE ###` indicates where you should write code. Be sure to replace this with your own code before running the code cell.
- Feel free to open the hints for additional guidance as you work on each task.
- To enter your answer to a question, double-click the markdown cell to edit. Be sure to replace the "[Double-click to enter your responses here.]" with your own answer.
- You can save your work manually by clicking File and then Save in the menu bar at the top of the notebook.
- You can download your work locally by clicking File and then Download and then specifying your preferred file format in the menu bar at the top of the notebook.

1.2 Scenario

In this lab, you're working as a security analyst and you're responsible for developing an algorithm that parses a file containing IP addresses that are allowed to access restricted content and removes addresses that no longer have access.

1.3 Task 1

Your eventual goal is to develop an algorithm that parses a series of IP addresses that can access restricted information and removes the addresses that are no longer allowed. Python can automate this process.

You're given a text file called `"allow_list.txt"` that contains a series of IP addresses that are allowed to access restricted information.

There are IP addresses that should no longer have access to this information, and their IP addresses need to be removed from the text file. You're given a variable named `remove_list` that contains the list of IP addresses to be removed.

Display both variables to explore their contents, and run the cell. Be sure to replace each `### YOUR CODE HERE ###` with your own code before running the following cell.

```
[1]: # 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"]

     # Display `import_file`

     print(import_file)

     # Display `remove_list`

     print(remove_list)
```

```
allow_list.txt
['192.168.97.225', '192.168.158.170', '192.168.201.40', '192.168.58.57']
```

Hint 1

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

Question 1   What do you observe about the output above?

IP addresses that are no longer allowed to access restricted information.

1.4   Task 2

In this task, start by opening the text file using the `import_file` variable, the `with` keyword, and the `open()` function with the `"r"` parameter. Be sure to replace the `### YOUR CODE HERE ###` with your own code.

For now, you'll 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; you'll complete this `with` statement in the task after this.

```
[ ]: # 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:
```

Hint 1

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.

Pass in the letter `"r"` as the second parameter when you want to read the file.

1.5 Task 3

Now, 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.

Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[2]: # 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
```

Hint 1

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

Hint 2

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

Hint 3

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

Question 2   Do you notice any IP addresses in the allow list that are also in the `remove_list`?

Yes, there are IP addresses that are also in the remove_list

1.6   Task 4

After reading the file, reassign the `ip_addresses` variable so its data type is updated from a string to a list. Use the `.split()` method to achieve this. Adding this step will allow you 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.

Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[3]: # 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']
```

Hint 1

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. Note that whitespace includes any space between text on the same line and the space between one line and the next line.

In this task, the default behavior of `.split()` works well. Each IP address is on a new line in the `allow_list.txt` file. In other words, there is whitespace between IP addresses in the text file. When you use `.split()`, it will separate the IP addresses and output them as a list.

Hint 2

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

1.7 Task 5

Now, you'll 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. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[7]: # 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
```

Hint 1

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

Hint 2

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

1.8 Task 6

Now, 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`. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[8]:  # 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
```

```python
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']
```

Hint 1

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

Hint 2

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

Hint 3

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

1.9 Task 7

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 its 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. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell. This code cell will not produce an output.

```
[15]:  # 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:
```

```python
    # 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)
```

Hint 1

To 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.

Hint 2

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

Hint 3

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

1.10   Task 8

In this task, you'll verify that the original file was rewritten using the correct list.

Write another `with` statement, this time to read in the updated file. Start by opening the file. Then read the file and store its contents in the `text` variable.

Afterwards, display the `text` variable to examine the result.

Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[18]: # 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)

# 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)
```

Hint 1

To 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 `"r"` as the second parameter.

The `"r"` parameter specifies that you're opening the file for the purpose of reading it.

Hint 2

Inside the `with` statement, call the `.read()` method to read the contents of the file. Assign the `text` variable to the result.

Hint 3

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

1.11   Task 9

The next step is to bring all of the code you've written leading up to this point and put it all into one function.

Define a function named `update_file()` that takes in two parameters. The first parameter is the name of the text file that contains IP addresses (call this parameter `import_file`). The second parameter is a list that contains IP addresses to be removed (call this parameter `remove_list`).

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell. Note that this code cell will not produce an output.

```
[19]:  # 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)
```

Hint 1

Use the `def` keyword to start the function definition.

Hint 2

After the `def` keyword, specify the name of the function, followed by parantheses and a colon.

Inside the parantheses, specify the parameters that the function takes in.

Hint 3

After the `def` keyword, write `update_file(import_file, remove_list):` to complete the function definition header.

Question 3   What are the benefits of incorporating the algorithm into a single function?

So that I can call this function in another part of my code

1.12   Task 10

Finally, call the `update_file()` that you defined. Apply the function to `"allow_list.txt"` and pass in a list of IP addresses as the second argument.

Use the following list of IP addresses as the second argument:

`["192.168.25.60", "192.168.140.81", "192.168.203.198"]`

After the function call, use a `with` statement to read the contents of the allow list. Then display the contents of the allow list. Run it to verify that the file has been updated by the function.

Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

```
[22]:  # 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`
```

```python
    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)

# Call `update_file()` and pass in "allow_list.txt" and a list of IP addresses
↪to be removed

update_file("allow_list.txt", ["192.168.25.60", "192.168.140.81", "192.168.203.
↪198"])

# Build `with` statement to read in the updated file

with open("allow_list.txt", "r") as file:

    # Read in the updated file and store the contents in `text`

    text = file.read()

# Display the contents of `text`

print(text)
```

Hint 1

To call the `update_file()` function, write the name of the function, followed by parantheses, and pass in the file name and list of IP addresses that you want to try out the function on. Be sure to

separate the two arguments with a comma (,).

Hint 2

Inside the `with` statement, call the `.read()` method to read the contents of the file. Assign the `text` variable to the result.

Hint 3

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

1.13   Conclusion

What are your key takeaways from this lab?

Algorithms help parse text file of IP addresses and updates the file by removing that addresses that no longer have access to the restricted content.