Activity_Create another algorithm

August 4, 2023

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 the addresses that no longer have access to the restricted content.

Note: *Have you already completed this lab once?* Due to how Coursera handles files, you will need to reset the file used in this lab to its original contents if you want to complete this lab more than once. The Section **??** section at the end of this notebook contains code that allows you to reset the `allow_list.txt"` file to its original contents. After you have run the code in that section, you can begin the lab again.

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.

[17]:
```
# 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?

I observed from the output that the `import_file` returned the name of the text file `"allow_list.txt"`, and the `remove_list` returned the list of IP addresses‘.

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.

[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"]

# First line of `with` statement

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

```
  File "<ipython-input-18-b925af1022fc>", line 11
    with open(import_file, "r") as file:
                                        ^
SyntaxError: unexpected EOF while parsing
```

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 in as a string 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.

[19]:
```
# 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`?

Upon review, I noticed that the 4 removed IP addresses from the `remove_list` variable are also in the allow list file, `"allow_list.txt"`.

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. Converting the string into a list will later allow you to remove individual IP addresses from the allow list.

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.

[20]:
```
# 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 using an iterative statement.

First, build the iterative statement. Name the loop variable `element` and loop through `remove_list`. In the body of the loop, display each element. (Displaying each element is a temporary step for this task.)

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

[23]:
```
# 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 `remove_list`

for element in remove_list:

  # Display `element` in every iteration

    print(element)
```

```
192.168.97.225
192.168.158.170
192.168.201.40
192.168.58.57
```

Hint 1

Build a `for` loop to iterate through `remove_list`. 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, you will update the body of the loop with code that will remove IP addresses from the allow list.

First, a conditional statement is used to evaluate if the loop variable `element` is part of the `ip_addresses` list. This is important to avoid the errors that would occur when using the `.remove()` method if an `element` was not part of the `ip_addresses` list.

Then, within the conditional, apply the `.remove()` method to the `ip_addresses` list and remove the IP addresses identified in the loop variable `element`.

After the iterative statement removes the elements, 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.

Note: There are not any duplicate values in the `ip_addresses` list. The `.remove()` method only removes the first occurrence of an element, so if there were duplicates, they would not be removed.

[24]:
```
# 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 `remove_list`

for element in remove_list:

    # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

        # use the `.remove()` method to remove
        # elements 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

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

Hint 2

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 `"\n"`. The `"\n"` character indicates to separate each element by placing it on a new line. 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 each IP address on a new line.

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.

[27]:
```
# 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 `remove_list`

for element in remove_list:

    # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

        # use the `.remove()` method to remove
        # elements 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 = "\n".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.

[37]:
```
# Assign `import_file` to the name of the file
# ET comment: I had to use 'data/allow_list.txt' from the exemplar to get the␣
 ↪output to work. Something is wrong with Task #8.

import_file = "data/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 `remove_list`

for element in remove_list:

    # Create conditional statement to evaluate if `element` is in `ip_addresses`

    if element in ip_addresses:

        # use the `.remove()` method to remove
        # elements 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 = "\n".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)
```

```
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

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.

[39]:
```
# 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 `remove_list`

    for element in remove_list:

        # Create conditional statement to evaluate if `element` is in␣
↪`ip_addresses`

        if element in ip_addresses:
```

```
            # use the `.remove()` method to remove
            # elements 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 = "\n".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?

The benefits of incorporating the algorithm into a single function ensures that the code maintains its readability and allows for further collaboration. It also allows the algorithm to be called once to run the multi line code we wrote. This will benefit by increasing efficiency and productivity at an organization.

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", "192.168.90.124", "192.168.60.153"]`

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.

[43]:
```
# 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 `remove_list`

    for element in remove_list:

        # Create conditional statement to evaluate if `element` is in␣
↪`ip_addresses`

        if element in ip_addresses:

            # use the `.remove()` method to remove
            # elements 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 = "\n".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
# ET comment: I had to use 'data/allow_list.txt' from the exemplar to get the␣
 ↪output to work. Something is wrong with Task #10.

update_file("data/allow_list.txt", ["192.168.25.60", "192.168.90.124", "192.168.
 ↪60.153"])

# Build `with` statement to read in the updated file
# ET comment: I had to use 'data/allow_list.txt' from the exemplar to get the␣
 ↪output to work. Something is wrong with Task #10.

with open("data/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)
```

```
ip
address
192.168.205.12
192.168.6.9
192.168.52.90
192.168.186.176
192.168.133.188
192.168.203.198
192.168.218.219
192.168.52.37
192.168.156.224
192.168.69.116
```

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?

My key takeaway from this lab is that Python is a great tool to automate tasks. I practiced more with the `open()` function to open files for reading (`"r"`) and writing (`"w"`). I also got more pracitce using the `.read()` function to read an open file, and the `.write()` function to add to the file. Another function that I got pracitce with from this lab was the `.split()` function. This function allowed me to convert strings to lists to allow for easier manipulation of the file. And, lastly and most importantly for algorithms/automation, I became more experienced with the iterative statments in Python. I created a `for` loop with `if` conditional statements. Perhaps to best keytakeaway from this lab is my confidence. I feel much more experienced in Python and confident that even if I cannot solve a task, I know how to break down the problem to find a solution and research in the correct ways to write effective code.

1.14 File contents reset

You can run the following code to reset the `"allow_list.txt"` file to its original contents. Because of how Coursera handles files, this will be necessary if you wish to complete this lab more than once or if you have unintentionally changed the file in a way that does not correspond to the lab tasks.

[15]:
```
# Resets the `"allow_list.txt"` file to its original contents
# Allows learners to complete lab more than once

# Assigns the original list of IP addresses to the `ip_addresses` variable

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
"""

# Writes `ip_addresses` to the `"allow_list.txt"` file

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