Activity_Create another algorithm

April 11, 2024

1 Activity: Create another algorithm

1.1 Introduction

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

1.2 Scenario

In this lab, I'm working as a security analyst and am 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

My 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. I'm 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. I am given a variable named `remove_list` that contains the list of IP addresses to be removed.

[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']
```

1.4 Task

In this task, I'll start by opening the text file using the `import_file` variable, the `with` keyword, and the `open()` function with the `"r"` parameter.

[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"]

# First line of `with` statement

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

```
  File "<ipython-input-2-a8b4359caa1c>", line 11
    with open (import_file, "r") as file:
                                         ^
SyntaxError: unexpected EOF while parsing
```

1.5 Task

Now, I'll use the `.read()` method to read the imported file and store it in a variable named `ip_addresses`.

[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()

# 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
```

1.6 Task

After reading the file, I have to reassign the `ip_addresses` variable so its data type is updated from a string to a list. I'll use the `.split()` method to achieve this. Adding this step will allow me 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.

[4]:
```
# 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']
```

1.7 Task

Now, I'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, I'll build the iterative statement, name the loop variable `element`, loop through `ip_addresses`, and display each element.

[5]:
```
# 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
```

1.8 Task

Now, I 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.

[9]:
```
# 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']
```

1.9 Task

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, I'll build the `with` statement that rewrites the original file. I'll use the `"w"` parameter when calling the `open()` function to delete the contents in the original file and replace it.

[10]:
```
# 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 (ip_addresses, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

1.10 Task

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

I'll write another `with` statement, this time to read in the updated file, then start by opening the file, read it and store its contents in the `text` variable.

[11]:
```
# 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(ip_addresses, "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
```

1.11 Task

The next step is to bring all of the code written and put it all into one function.

I'll 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`).

[12]:
```
# 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)
```

1.12 Task

Finally, I'll now call the `update_file()` that I defined. I'll apply the function to `"allow_list.txt"` and pass in a list of IP addresses as the second argument.

I'll 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, I'll use a `with` statement to read the contents of the allow list. Then display the contents of the allow list.

[15]:
```
# 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)

# 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)
```

```
ip_address 192.168.205.12 192.168.6.9 192.168.52.90 192.168.90.124
192.168.186.176 192.168.133.188 192.168.218.219 192.168.52.37 192.168.156.224
192.168.60.153 192.168.69.116
```

1.13 Conclusion

What are your key takeaways from this lab?

[- Python has functions and syntax that help one import and parse text files. - The `with` statement allows one to efficiently handle files. - The `open()` function allows one 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. - The `.read()` method allows one to read in a file. - The `.write()` method allows one to append or write to a file. - One can use Python to compare contents of a text file against elements of a list. - Algorithms can be incorporated into functions.