Activity: Import and parse a text file

Introduction ¶

Security logs are often stored in text files. To analyze the security logs in these files, security analysts have to import and parse these files. Python has some functions that come in handy for these tasks, allowing analysts to efficiently access information from text files.

In this lab, you'll practice using functions and other syntax in Python to import and parse text files.

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

Scenario

In this lab, you're working as a security analyst. You're responsible for preparing a security log file for analysis and creating a text file with IP addresses that are allowed to access restricted information.

Task 1

In this task, you'll import a security log text file and store it as a string to prepare it for analysis.

In Python, a `with` statement is often used in file handling to open a file and then automatically close the file after reading it.

You're given a variable named `import_file` that contains the name of the log file that you want to import. Start by writing the first line of the `with` statement in the following code cell. Use the `open()` function, setting the second parameter to `"r"` . Note that 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. Be sure to replace the `### YOUR CODE HERE ###` with your own code.

In [1]:
```
# Assign `import_file` to the name of the text file that contains the security log file
import_file = "login.txt"

# First line of the `with` statement
# Use `open()` to import security log file and store it as a string
with open(import_file, "r") as file:
    # Rest of the code goes here
    # You can now analyze or process the contents of the log file within this block
    content = file.read()

# Example: Printing the content of the log file
print(content)
```

```
username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09
```

Hint 1

Hint 2

Task 2

Now, you'll use the `.read()` method to read the imported file, and you'll store the result in a variable named `text`. Afterwards, display the `text` and explore what it contains by running the cell. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

In [2]:
```
# Assign `import_file` to the name of the text file that contains the security log file
import_file = "login.txt"

# The `with` statement
# Use `open()` to import security log file and store it as a string
with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store the result in a variable named `text`
  text = file.read()

# Display the contents of `text`
print(text)

# This code opens the "login.txt" file, reads its content using .read(), and stores the result in the text variable. Finally, it prints the content of the text variable, allowing you to explore what the log file contains.
```

```
username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09
```

Hint 1

Hint 2

Hint 3

Task 3

The output in the previous step is one big string. In this task, you'll explore how you can split the string that contains the entire imported log file into a list of strings, one string per line.

Use the `.split()` method to perform this split and then display the result. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

Note that displaying `.split()` doesn't change what is stored in the `text` variable. Variable reassignment would be necessary if you want to store the result after splitting.

In [3]:
```
# Assign `import_file` to the name of the text file that contains the security log file
import_file = "login.txt"

# The `with` statement
# Use `open()` to import security log file and store it as a string
with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store the result in a variable named `text`
  text = file.read()

# Display the contents of `text` split into separate lines
lines = text.split('\n')
print(lines)

#This code reads the content of the file into the text variable and then uses .split('\n') to split the text into a list of strings, where each string corresponds to a line in the log file. Finally, it prints the resulting list, displaying each line of the log file as a separate element.
```

```
['username,ip_address,time,date', 'tshah,192.168.92.147,15:26:08,2022-05-10',
'dtanaka,192.168.98.221,9:45:18,2022-05-09', 'tmitchel,192.168.110.131,14:13:
41,2022-05-11', 'daquino,192.168.168.144,7:02:35,2022-05-08', 'eraab,192.168.
170.243,1:45:14,2022-05-11', 'jlansky,192.168.238.42,1:07:11,2022-05-11', 'ac
ook,192.168.52.90,9:56:48,2022-05-10', 'asundara,192.168.58.217,23:17:52,2022
-05-12', 'jclark,192.168.214.49,20:49:00,2022-05-10', 'cjackson,192.168.247.1
53,19:36:42,2022-05-12', 'jclark,192.168.197.247,14:11:04,2022-05-12', 'apate
l,192.168.46.207,17:39:42,2022-05-10', 'mabadi,192.168.96.244,10:24:43,2022-0
5-12', 'iuduike,192.168.131.147,17:50:00,2022-05-11', 'abellmas,192.168.60.11
1,13:37:05,2022-05-10', 'gesparza,192.168.148.80,6:30:14,2022-05-11', 'cgriff
in,192.168.4.157,23:04:05,2022-05-09', 'alevitsk,192.168.210.228,8:10:43,2022
-05-08', 'eraab,192.168.24.12,11:29:27,2022-05-11', 'jsoto,192.168.25.60,5:0
9:21,2022-05-09', '']
```

Hint 1

Hint 2

Question 1

What do you notice about the output before and after using the `.split()` method?

Before using the .split() method, the output is a single string containing the entire content of the log file, with line breaks represented by '\n' characters. After using the .split() method, the output is a list of strings, where each string corresponds to a line from the original log file.

The .split('\n') method splits the original string based on newline characters, creating a list of strings where each element represents a line in the log file. This can be useful for further processing or analyzing each line separately.

Task 4

There is a missing entry in the log file. You'll need to account for that by appending it to the log file. You're given the missing entry stored in a variable named `missing_entry`.

Use the `.write()` method and the parameter `"a"` in the `open()` function. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

After the portion of the code that writes to the file, another with statement uses the `.read()` method to read the updated file into the `text` variable and then display it.

In [4]:
```
# Assign `import_file` to the name of the text file that contains the security log file
import_file = "login.txt"

# Assign `missing entry` to a log that was not recorded in the log file
missing_entry = "jrafael,192.168.243.140,4:56:27,2022-05-09"

# Use `open()` to import security log file and store it as a string
# Pass in "a" as the second parameter to indicate that the file is being opened for appending purposes
with open(import_file, "a") as file:

    # Use `.write()` to append `missing_entry` to the log file
    file.write("\n" + missing_entry)

# Use `open()` with the parameter "r" to open the security log file for reading purposes
with open(import_file, "r") as file:

    # Use `.read()` to read in the contents of the log file and store in a variable named `text`
    text = file.read()

# Display the contents of `text`
print(text)

# This code appends the missing_entry to the log file using the .write() method and then reads the updated file into the text variable using .read(). Finally, it prints the updated content to verify the addition of the missing entry.
```

```
username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09

jrafael,192.168.243.140,4:56:27,2022-05-09
```

Hint 1

Hint 2

Hint 3

Question 2

What do you notice about the position of the entry that was added to the log file?

The added entry is positioned at the end of the log file. This is because the code uses the "a" (append) mode when opening the file with open(import_file, "a"), which allows new data to be written at the end of the file, maintaining the existing content and adding the new entry below it in the code block.

Task 5

The next task you're responsible for is creating a text file. This text file should include a list of IP addresses that are allowed to access restricted information. Documenting this in a text file will help you communicate your findings to your security team.

Start by creating a variable named `import_file` that stores the name of the file, which should be `"allow_list.txt"` .

You're also given a variable named `ip_addresses` that stores a string containing the IP addresses that are allowed.

Run the code to display the two variables and explore what they contain. Be sure to replace the `### YOUR CODE HERE ###` with your own code before you run the following cell.

In [5]:
```
# Assign `import_file` to the name of the text file that you want to create
import_file = "allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access t
he restricted information
ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13
192.168.60.153 192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 19
2.168.15.110 192.168.39.246"

# Display `import_file`
print("File Name:", import_file)

# Display `ip_addresses`
print("IP Addresses:")
print(ip_addresses)

# This code assigns the name "allow_list.txt" to the import_file variable and
a string containing a list of IP addresses to the ip_addresses variable. The p
rint statements display the content of these variables for exploration.
```

```
File Name: allow_list.txt
IP Addresses:
192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13 192.168.60.153
192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 192.168.15.110 1
92.168.39.246
```

Hint 1

Task 6

Your next goal is to create a `with` statement in order to write the IP addresses to the text file you created in the previous step.

You'll first open the file using the `"w"` parameter. Then, you'll write the IP addresses to the file. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell. Note that the code cell will contain a `with` statement that writes to a file but does not display information to the screen, so running it will not produce an output.

In [7]:
```
# Assign `import_file` to the name of the text file that you want to create
import_file = "allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access t
he restricted information
ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13
192.168.60.153 192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 19
2.168.15.110 192.168.39.246"

# Create a `with` statement to write to the text file
with open(import_file, "w") as file:

  # Write `ip_addresses` to the text file
  file.write(ip_addresses)

# This code uses the with statement to open the "allow_list.txt" file in write
mode ("w") and writes the content of the ip_addresses variable to the file. Th
e file is automatically closed after writing.
```

Hint 1

Hint 2

Hint 3

Task 7

In this final step, you'll complete the code you've been writing up to this point. You'll add code to read the file containing IP addresses.

Complete a `with` statement that reads the text file and stores it in a new variable called `text`.

Afterwards, display the contents of `text` and run the cell to explore the result. Be sure to replace each `### YOUR CODE HERE ###` with your own code before you run the following cell.

In [8]:
```
# Assign `import_file` to the name of the text file that you want to create
import_file = "allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access t
he restricted information
ip_addresses = "192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13
192.168.60.153 192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 19
2.168.15.110 192.168.39.246"

# Create a `with` statement to write to the text file
with open(import_file, "w") as file:
    # Write `ip_addresses` to the text file
    file.write(ip_addresses)

# Create a `with` statement to read in the text file
with open(import_file, "r") as file:
    # Read the file and store the result in a variable named `text`
    text = file.read()

# Display the contents of `text`
print(text)

# This code first writes the ip_addresses to the "allow_list.txt" file using a
with statement with write mode ("w"). Then, it uses another with statement to
read the content of the file into the variable text with read mode ("r"). Fina
lly, it prints the content of text to explore the result.
```

```
192.168.218.160 192.168.97.225 192.168.145.158 192.168.108.13 192.168.60.153
192.168.96.200 192.168.247.153 192.168.3.252 192.168.116.187 192.168.15.110 1
92.168.39.246
```

Hint 1

Hint 2

Hint 3

Conclusion

What are your key takeaways from this lab?

The key takeaways from this lab are:

File Handling in Python: Understanding how to use the with statement and the open() function to handle files, both for reading ("r") and writing ("w").

Reading and Writing Text Files: Using the .read() method to read the contents of a file into a string and the .write() method to write data to a file.

Appending to Files: Using the "a" mode with open() to append data to the end of an existing file.

Splitting Strings: Using the .split() method to split a string into a list of strings based on a specified delimiter, such as a newline character.

Creating and Writing to a New Text File: Creating a new text file and writing data to it.

Understanding Modes in open(): Modes like "r" (read), "w" (write), and "a" (append) control the behavior of the open() function.

These concepts are fundamental for handling files in Python and are applicable in various scenarios, such as reading and processing log files, creating configuration files, or managing data storage.

File contents reset

You can run the following code to reset the `"login.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.

In [10]:
```
# Resets the `"login.txt"` file to its original contents
# Allows learners to complete lab more than once

# Assigns the original contents of the file to the `login_file` variable
login_file = """username,ip_address,time,date
tshah,192.168.92.147,15:26:08,2022-05-10
dtanaka,192.168.98.221,9:45:18,2022-05-09
tmitchel,192.168.110.131,14:13:41,2022-05-11
daquino,192.168.168.144,7:02:35,2022-05-08
eraab,192.168.170.243,1:45:14,2022-05-11
jlansky,192.168.238.42,1:07:11,2022-05-11
acook,192.168.52.90,9:56:48,2022-05-10
asundara,192.168.58.217,23:17:52,2022-05-12
jclark,192.168.214.49,20:49:00,2022-05-10
cjackson,192.168.247.153,19:36:42,2022-05-12
jclark,192.168.197.247,14:11:04,2022-05-12
apatel,192.168.46.207,17:39:42,2022-05-10
mabadi,192.168.96.244,10:24:43,2022-05-12
iuduike,192.168.131.147,17:50:00,2022-05-11
abellmas,192.168.60.111,13:37:05,2022-05-10
gesparza,192.168.148.80,6:30:14,2022-05-11
cgriffin,192.168.4.157,23:04:05,2022-05-09
alevitsk,192.168.210.228,8:10:43,2022-05-08
eraab,192.168.24.12,11:29:27,2022-05-11
jsoto,192.168.25.60,5:09:21,2022-05-09
"""

# Writes `login_file` to the `"login.txt"` file
with open("login.txt", "w") as file:
          file.write(login_file)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: