Exemplar_Import and parse a text file

February 5, 2024

1 Exemplar_Import and parse a text file

1.1 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 `data/login.txt"` 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 `data/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

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. 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.

1.3 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.

[ ]:
```
# Assign `import_file` to the name of the text file that contains the security␣
↪log file

import_file = "data/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:
```

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 argument when you want to read the file.

Hint 2

The `import_file` variable contains the name of the file that you want to open, so that should be the first argument you pass in to the `open()` function.

Since you also want to read the contents of the file, you should pass in `"r"` as the second argument.

Make sure to use a comma (`,`) to separate the two arguments.

1.4 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.

[ ]:
```
# Assign `import_file` to the name of the text file that contains the security␣
↪log file

```

```
import_file = "data/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)
```

Hint 1

The `.read()` method in Python converts text files to strings.

Hint 2

The `file` object contains the file that you want to read, so apply the `.read()` method to `file`.

Hint 3

Use the `print()` function to display the contents of `text`.

1.5 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.

[ ]:
```
# Assign `import_file` to the name of the text file that contains the security␣
 ↪log file

import_file = "data/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

print(text.split())
```

Hint 1

The `.split()` method in Python converts a string into a list. It can take in a separator character that specifies which character to split on. If a character is not specified, it will split on whitespace by default. This default will work well for your task, since the log file contains whitespace between each line in the log.

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

Hint 2

Use the `.split()` method to convert the `text` into a list, where each element in the list represents a line in the log file.

Place this between the parantheses in the `print()` function call.

Question 1 What do you notice about the output before and after using the `.split()` method?

Before using the `.split()` method, the output is one long string containing all of the lines from the log file. After using the `.split()` method, the output is a list of strings; each string corresponds to a line from the log file.

1.6 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.

[ ]:
```
# Assign `import_file` to the name of the text file that contains the security␣
 ↪log file

import_file = "data/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(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)
```

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 `"a"` as the second parameter when you want to append the file.

Hint 2

Call the `.write()` method on the log file and pass in `missing_entry`. This will append `missing_entry` to the log file.

Hint 3

Call `file.write()` and pass in `missing_entry`. This will append `missing_entry` to the log file.

Question 2 What do you notice about the position of the entry that was added to the log file?

The additional entry was added to the end of the log file, so it appears in the last line of the output.

1.7 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.

[ ]:
```
# Assign `import_file` to the name of the text file that you want to create

import_file = "data/allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access␣
 ↪the 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␣
 ↪192.168.15.110 192.168.39.246"

# Display `import_file`

print(import_file)

# Display `ip_addresses`

print(ip_addresses)
```

Hint 1

Keep in mind that the name of the text file you want to create should be `"allow_list.txt"`. Make sure to include the `.txt` file extension, which specifies the file format.

1.8 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.

```
[ ]: # Assign `import_file` to the name of the text file that you want to create

import_file = "data/allow_list.txt"

# Assign `ip_addresses` to a list of IP addresses that are allowed to access␣
 ↪the 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␣
 ↪192.168.15.110 192.168.39.246"

# 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)
```

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 `"w"` as the second parameter when you're opening a file for the purpose of writing to it.

Hint 2

Call the `.write()` method on the text file to write to it.

Hint 3

Call the `file.write()` method and pass in the `ip_addresses` variable to write the contents of that variable to the text file.

1.9 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.

```
[ ]: # Assign `import_file` to the name of the text file that you want to create

import_file = "data/allow_list.txt"
```

```
# Assign `ip_addresses` to a list of IP addresses that are allowed to access␣
 ↪the 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␣
 ↪192.168.15.110 192.168.39.246"

# 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)
```

Hint 1

The `open()` function in Python allows you to open a file.

It takes in the name of the file as the first parameter and a string that indicates how the file should be handled as the second parameter.

Pass in the letter `"r"` as the second parameter when you're opening a file for the purpose of reading in its contents.

Hint 2

Call the `.read()` method on the text file to read it in.

Hint 3

Call `file.read()`. Place this to the right of the `=` operator to assign the output to the `text` variable.

1.10 Conclusion

What are your key takeaways from this lab?

- 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 `"a"` as the second parameter if you're opening the file for appending 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.
- The `.split()` method in Python allows you to convert a string to a list.

1.11 File contents reset

You can run the following code to reset the `"data/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.

[ ]:
```
# Resets the `"data/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 `"data/login.txt"` file
with open("data/login.txt", "w") as file:
          file.write(login_file)
```