Working with Regular Expressions

1 hour 30 minutes Free ★★★★☆

Introduction

It's time to put your new skills to the test! In this lab, you'll have to find the users using an old email domain in a big list using regular expressions. To do so, you'll need to write a script that includes:

- Replacing the old domain name (abc.edu) with a new domain name (xyz.edu).
- Storing all domain names, including the updated ones, in a new file.

You'll have 90 minutes to complete this lab.

Prerequisites

We've created a list containing user names and their email addresses. Navigate to the **data** directory using the following command:

```
cd data
```

To find the data, list the files using the following command:

```
ls
```

You can now see a file named **user_emails.csv**. This is where you will find the required data. To view the contents of the **user_emails.csv** file, enter the following command:

```
cat user_emails.csv
```

You can also access a python script that contains function definitions for the task. Navigate to the **scripts** directory using the following command:

```
cd ~/scripts
```

Now list the contents within the **scripts** directory using the following command:

Now list the contents within the **scripts** directory using the following command:

```
ls
```

Here, you will find a file named **script.py**. The aim of this script is to use regex to find all instances of the old domain ("abc.edu") in the **user_emails.csv** file and then replace them with the new domain ("xyz.edu").

This file already has the functions defined for you. You have to now complete the function's body to make it work as intended.

Let's update the file's permissions.

```
sudo chmod 777 script.py
```

We will use nano editor to edit **script.py** file.

```
nano script.py
```

Before we start writing the script, let's import libraries to use in the script. To do this, open the file with nano editor. To deal with CSV file operations, Python has a CSV module that effectively handles CSV data. Let's import the CSV module using the following:

```
import csv
```

effectively handles CSV data. Let's import the CSV module using the following:

```
import csv
```

Import the regex Python module (i.e the regular expression module) to this script. A regular expression(RegEx) is a sequence of characters that defines a search pattern.

```
import re
```

Identify the old domain

In this section, we will write the body of the function named *contains_domain*. This function uses *regex* to identify the domain of the user email addresses in the user_emails.csv file.

The function takes address and domain as parameters, and its primary objective is to check whether an email address belongs to the old domain(abc.edu).

To do this, we will use a regular expression stored in the variable named *domain_pattern*. This variable will now match email addresses of a particular domain. If the old domain is found, then the function returns true.

Identify the old domain

In this section, we will write the body of the function named *contains_domain*. This function uses *regex* to identify the domain of the user email addresses in the user_emails.csv file.

The function takes address and domain as parameters, and its primary objective is to check whether an email address belongs to the old domain(abc.edu).

To do this, we will use a regular expression stored in the variable named *domain_pattern*. This variable will now match email addresses of a particular domain. If the old domain is found, then the function returns true.

```
  domain_pattern = r'[\w\.-]+@'+domain+'$'
  if re.match(domain_pattern, address):
    return True
```

The function *contains_domain* should now look like this:

```
def contains_domain(address, domain):
  domain_pattern = r'[\w\.-]+@'+domain+'$'
  if re.match(domain_pattern, address):
    return True
  return False
```

Replace the domain name

In this section, we will replace the old domain name with the new one. The second function defined in the **script.py** file is *replace_domain*.

The *replace_domain* function takes in one email address at a time, as well as the email's old domain name and its new domain name. This function's primary objective is to replace the email addresses containing the old domain name with new domain name.

In order to replace the domain name, we will use the regular expression module and make a pattern that identifies sub-strings containing the old domain name within email addresses. We will then store this pattern in a variable called *old_domain_pattern*. Next, we will use substitution function *sub()* from ***re*** module to replace the old domain name with the new one and return the updated email address.

```
  old_domain_pattern = r'' + old_domain + '$'
  address = re.sub(old_domain_pattern, new_domain, address)
```

The function *replace_domain* should now look similar to the following:

```
def replace_domain(address, old_domain, new_domain):
  old_domain_pattern = r'' + old_domain + '$'
  address = re.sub(old_domain_pattern, new_domain, address)
  return address
```

Write a CSV file with replaced domain from main

In this section, we're going to call the above defined functions: contains_domain() and replace_domain from the main(). This will allow us to find the old domain email address, replace it with the newer one, and write the updated list to a CSV file in the data directory.

In the previous sections, you might have seen variables named *old_domain* and *new_domain*, which are passed as parameters to the functions. Let's declare them here within main().

```
old_domain, new_domain = 'abc.edu', 'xyz.edu'
```

Now store the path of the list **user_emails.csv** in the variable *csv_file_location*. Also, give a file path for the resulting updated list within the variable *report_file*. This updated list should be generated within the **data** directory.

```
csv_file_location = '<csv-file-location>'
report_file =  '<data-directory>' +
'/updated_user_emails.csv'
```

Replace `<csv_file_location>` by the path to the user_emails.csv. `<csv_file_location>` is similar to the path */home/<username>/data/user_emails.csv*. For variable report_file, replace <data_directory> by the path to */data* directory. `<data_directory>` is similar to the path */home/<username>/data*. Replace

Replace `<csv_file_location>` by the path to the user_emails.csv. `<csv_file_location>` is similar to the path */home/<username>/data/user_emails.csv*. For variable report_file, replace <data_directory> by the path to */data* directory. `<data_directory>` is similar to the path */home/<username>/data*. Replace `<username>` with the one mentioned in the Connection Details Panel on the left-hand side.

Then, initialize an empty list where you will store the user email addresses. This is then passed to the function *contains_domain*, where a regular expression is used to match them and finally replace the domains using the *replace_domain* function.

Next, initialize the two different lists, **old_domain_email_list** and **new_domain_email_list**. The **old_domain_email_list** will contain all the email addresses with the old domain that the regex would match within the function **contains_domain**. Since the function *contains_domain* takes in email address passed as parameter, we will iterate over the **user_email_list** to pass email addresses one by one. For every matched email address, we will append it to the list old_domain_email_list.

```
user_email_list = []
old_domain_email_list = []
new_domain_email_list = []
```

The CSV module imported earlier implements classes to read and write tabular data in CSV format. The CSV library provides functionality to both read from and write to CSV files. In this case, we are first going to read data from the list (which is a CSV file). The data is read from the **user_emails.csv** file and passed to the **user_data_list**. So the **user_data_list** now contains the same information as that present in **user_emails.csv** file. While we do this, we will also add all the email addresses into the **user_email_list** that we

user_data_list now contains the same information as that present in **user_emails.csv** file. While we do this, we will also add all the email addresses into the **user_email_list** that we initialized in the previous step.

```
  with open(csv_file_location, 'r') as f:
    user_data_list = list(csv.reader(f))
    user_email_list = [data[1].strip() for data in
user_data_list[1:]]
```

The list **old_domain_email_list** should contain all the email addresses with the old domain. This will be checked by the function *contains_domain*. The function *replace_domain* will then take in the email addresses (with old domain) and replace them with the new domains.

```
    for email_address in user_email_list:
      if contains_domain(email_address, old_domain):
        old_domain_email_list.append(email_address)
        replaced_email = replace_domain(email_address,
old_domain, new_domain)
        new_domain_email_list.append(replaced_email)
```

Now, let's define the headers for our output file through the **user_data_list**, which contains all the data read from user_emails.csv file.

```
    email_key = ' ' + 'Email Address'
    email_index = user_data_list[0].index(email_key)
```


```
    email_key = ' ' + 'Email Address'
    email_index = user_data_list[0].index(email_key)
```


Next, replace the email addresses within the **user_data_list** (which initially had all the user names and respective email addresses read from the user_emails.csv file) by iterating over the **new_domain_email_list**, and replacing the corresponding values in **user_data_list**.

Finally, close the file using the close() method. A closed file no longer be read or written. It is good practice to use the close() method to close a file.

```
    for user in user_data_list[1:]:
      for old_domain, new_domain in zip(old_domain_email_list,
new_domain_email_list):
        if user[email_index] == ' ' + old_domain:
          user[email_index] = ' ' + new_domain
    f.close()
```

Now write the list to an output file, which we declared at the beginning of the script within the variable **report_file**.

```
  with open(report_file, 'w+') as output_file:
    writer = csv.writer(output_file)
    writer.writerows(user_data_list)
    output_file.close()
```

Finally, call the main() method.

Finally, call the main() method.

```
main()
```

The script should now look like this:

```
#!/usr/bin/env python3
import re
import csv
def contains_domain(address, domain):
  """Returns True if the email address contains the
given,domain,in the domain position, false if not."""
  domain = r'[\w\.-]+@'+domain+'$'
  if re.match(domain,address):
    return True
  return False
def replace_domain(address, old_domain, new_domain):
  """Replaces the old domain with the new domain in the
received address."""
  old_domain_pattern = r'' + old_domain + '$'
  address = re.sub(old_domain_pattern, new_domain, address)
  return address
def main():
  """Processes the list of emails, replacing any instances of
the old domain with the new domain."""
  old_domain, new_domain = 'abc.edu', 'xyz.edu'
  csv_file_location = '<csv_file_location>'
  report_file = '<path_to_home_directory>' +
'/updated_user_emails.csv'
  user_email_list = []
  old_domain_email_list = []
  new_domain_email_list = []
  with open(csv_file_location, 'r') as f:
    user_data_list = list(csv.reader(f))
    user_email_list = [data[1].strip() for data in
user_data_list[1:]]
    for email_address in user_email_list:
      if contains_domain(email_address, old_domain):
        old_domain_email_list.append(email_address)
        replaced_email =
replace_domain(email_address,old_domain,new_domain)
        new_domain_email_list.append(replaced_email)
    email_key = ' ' + 'Email Address'
    email_index = user_data_list[0].index(email_key)
    for user in user_data_list[1:]:
      for old_domain, new_domain in zip(old_domain_email_list,
new_domain_email_list):
        if user[email_index] == ' ' + old_domain:
          user[email_index] = ' ' + new_domain
  f.close()
  with open(report_file, 'w+') as output_file:
    writer = csv.writer(output_file)
    writer.writerows(user_data_list)
    output_file.close()
main()
```

Save the file by clicking **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Save the file by clicking **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Now run the file.

```
./script.py
```

On a successful run, this should generate a new file named **updated_user_emails** within the **data** directory.

To view the newly generated file, enter the following command:

```
ls ~/data
```

You should now be able to see a new file named **updated_user_emails.csv**. To view the contents of this file, enter the following command:

```
cat ~/data/updated_user_emails.csv
```

Great job! You have successfully replaced the old domain names with the new ones and generated a new file containing all the user names with their respective email addresses.

The report file should be similar to the one below image:

Click *Check my progress* to verify the objective.

Replace domain name

Check my progress

Congratulations!

You successfully wrote a Python script that achieves two tasks. First, it changed the domain name to the new domain name. Second, it stored all the updated domain names in a new file.

Creating reports using Python with CSV and using regular expressions to find a pattern in a string are very useful tools in IT support. You'll likely complete similar tasks regularly throughout your career, so feel free to go through this lab as many times as you need. Remember, practice makes perfect.

You can now close the RDP/SSH window. You can manually end the lab, or it will automatically end when the time runs out.