Fall 2018

Jupyter: Intro to Data Science - Assignment 2, Part 2 Details

Grant Long
CUNY City College

NYC Tech-in-Residence Corps

Assignment 2, Part 2 Details

Note: this is the second part of Assignment 2. Students are also required to complete the DataCamp course *Cleaning Data in Python* to receive full credit for Assignment 2.

For this short assignment, you'll be working to collect film permit data from NYC Open Data. Documentation for the API can be found at this link (https://data.cityofnewyork.us/City-Government/Film-Permits/tg4x-b46p). You should be able to complete all of the tasks outlined in this notebook without registering for an API key, but if not, you'll need to register for one and add it to your `params`.

Make sure to add your name to the `NAME` variable below before submitting your assignment.

In [ ]:
```
import pandas as pd
import requests

NAME = "Firstname Lastname"
```

Problem 1

Assign a string variable `url` with the web address for the base endpoint of the Film Permit Data, the documentation for which can be found here (https://dev.socrata.com/foundry/data.cityofnewyork.us/6aka-uima). This is the web address you will request *without* adding any parameters.

In [ ]:
```
#url = 'answer'

# YOUR CODE HERE
raise NotImplementedError()

url
```

Problem 2

Next, we'll want to create a a dictionary of values `params` to feed in the parameters for our API query. Let's create a set of parameters that will search for all `'Television'` permits in the zip code 10036, that will limit our results to 5000 permits. Again, you'll need to make use of the api documentation (https://dev.socrata.com/foundry/data.cityofnewyork.us/6aka-uima).

In [ ]:
```
# Hint: the limit parameter follows a diffent convention than the table
 fields

#params = {'field_name1' : 'Television',
#          'field_name2' : '10036',
#          'limit_parameter' : 5000}

# YOUR CODE HERE
raise NotImplementedError()

params
```

Problem 3

Next, use the `url` and `params` variables, along with the `requests` module to pull the data we're interested in from the API. Create a string variable `reponse_url` to equal the full url for the api call (using the `.url()`) and `response_json` to equal the full json contents of the response for the api call (using the

In [ ]:
```
# Hint: the strings listed below are guides to the correct answers (whic
h are not necessarily strings)

#response = 'requests.XXX(XXX=XXX, XXX=XXX)'

#response_url = 'response.XXX'
#response_json = 'response.XXX()'

# YOUR CODE HERE
raise NotImplementedError()

response_url
```

Problem 4

Transform the response into pandas DataFrame. Set `shape_result` equal to a tuple with the shape of the resulting DataFrame.

In [ ]:
```
# Hint: the strings listed below are guides to the correct answers (whic
h are not necessarily strings)

#df = 'pd.XXXX'
#shape_result = 'df.XXXX'

# YOUR CODE HERE
raise NotImplementedError()

shape_result
```

Problem 5

Now that the DataFrame is loaded, use the `.value_counts()` method to find the top sub-categories for television permits. Set `top_subcategory_counts` equal to the 5 top value counts for television permits, and `top_subcategory` equal to the name of the top sub-category for television permits.

In [ ]:
```
# Hint: the strings listed below are guides to the correct answers (whic
h are not necessarily strings)

#top_subcategory_counts = 'df.XXX.XXX().XXX(X)'
#top_subcategory = 'top_subcategory_counts.XXX[0]'

# YOUR CODE HERE
raise NotImplementedError()

top_subcategory
```

Problem 6

Finally, in the cell below, briefly describe one interesting statistic or data point you can calculate in the data, including the numerical value of the statistic, a short description of what it means, and a short explanation of why you find it interesting.

YOUR ANSWER HERE

In [ ]:
```

```