Data Science â€“ Learning Management Questions

Week Six â€“ Question #1

from pandas import \*

from ggplot import \*

def plot\_weather\_data(turnstile\_weather):

'''

You are passed in a dataframe called turnstile\_weather.

Use turnstile\_weather along with ggplot to make a data visualization

focused on the MTA and weather data we used in assignment #3.

You should feel free to implement something that we discussed in class

(e.g., scatterplots, line plots, or histograms) or attempt to implement

something more advanced if you'd like.

Here are some suggestions for things to investigate and illustrate:

\* Ridership by time of day or day of week

\* How ridership varies based on Subway station (UNIT)

\* Which stations have more exits or entries at different times of day

(You can use UNIT as a proxy for subway station.)

If you'd like to learn more about ggplot and its capabilities, take

a look at the documentation at:

https://pypi.python.org/pypi/ggplot/

You can check out:

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile\_data\_master\_with\_weather.csv

To see all the columns and data points included in the turnstile\_weather

dataframe.

However, due to the limitation of our Amazon EC2 server, we are giving you a random

subset, about 1/3 of the actual data in the turnstile\_weather dataframe.

'''

turnstile\_weather = turnstile\_weather[["ENTRIESn\_hourly"]].groupby(turnstile\_weather['UNIT']).mean()

turnstile\_weather=turnstile\_weather.reset\_index()

by\_num\_of\_ent = turnstile\_weather.sort\_values(["ENTRIESn\_hourly", "UNIT"], ascending=False)

by\_num\_of\_ent=by\_num\_of\_ent.head(15)

by\_num\_of\_ent=by\_num\_of\_ent.reset\_index()

plot = ggplot(by\_num\_of\_ent, aes('UNIT', 'ENTRIESn\_hourly')) + geom\_bar(aes(weight='ENTRIESn\_hourly'),fill='blue',stat = 'identity') + ggtitle('How ridership varies by subway station') + xlab('Station') + ylab('Entries')

return plot

if \_\_name\_\_ == "\_\_main\_\_":

filename = 'turnstile\_data\_master\_with\_weather.csv'

df=pandas.read\_csv(filename)

output=plot\_weather\_data(df)

print(output)

Week Six â€“ Question #2

from pandas import \*

from ggplot import \*

import datetime

def plot\_weather\_data(turnstile\_weather):

'''

plot\_weather\_data is passed a dataframe called turnstile\_weather.

Use turnstile\_weather along with ggplot to make another data visualization

focused on the MTA and weather data we used in Project 3.

Make a type of visualization different than what you did in the previous exercise.

Try to use the data in a different way (e.g., if you made a lineplot concerning

ridership and time of day in exercise #1, maybe look at weather and try to make a

histogram in this exercise). Or try to use multiple encodings in your graph if

you didn't in the previous exercise.

You should feel free to implement something that we discussed in class

(e.g., scatterplots, line plots, or histograms) or attempt to implement

something more advanced if you'd like.

Here are some suggestions for things to investigate and illustrate:

\* Ridership by time-of-day or day-of-week

\* How ridership varies by subway station (UNIT)

\* Which stations have more exits or entries at different times of day

(You can use UNIT as a proxy for subway station.)

If you'd like to learn more about ggplot and its capabilities, take

a look at the documentation at:

https://pypi.python.org/pypi/ggplot/

You can check out the link

https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/turnstile\_data\_master\_with\_weather.csv

to see all the columns and data points included in the turnstile\_weather

dataframe.

However, due to the limitation of our Amazon EC2 server, we are giving you a random

subset, about 1/3 of the actual data in the turnstile\_weather dataframe.

'''

turnstile\_weather = turnstile\_weather.copy()

f = '%Y-%m-%d'

turnstile\_weather['DATEn'] = turnstile\_weather['DATEn'].apply(lambda x: datetime.datetime.strptime(x, f))

turnstile\_weather['index'] = turnstile\_weather['DATEn'].dt.dayofweek

turnstile\_weather['DATEn'] = turnstile\_weather['DATEn'].apply(lambda x: x.strftime('%A'))

days\_df = turnstile\_weather[['DATEn','ENTRIESn\_hourly','index']].groupby('DATEn', as\_index=False).mean()

by\_ent = days\_df.sort\_values(['index',"ENTRIESn\_hourly","DATEn"], ascending=True)

plot = ggplot(by\_ent, aes(x='DATEn', y='ENTRIESn\_hourly')) + geom\_bar(aes(weight='ENTRIESn\_hourly'),fill='blue',stat = 'identity') + xlab('Day of week') +ylab('Entries per day') + ggtitle("Ridership by day of week")

return plot

if \_\_name\_\_ == "\_\_main\_\_":

filename = 'turnstile\_data\_master\_with\_weather.csv'

df=pandas.read\_csv(filename)

Week Six � Question #3  
import logging  
import sys  
import string  
  
logging.basicConfig(filename="loggger.log", format='%(message)s',  
 level=logging.INFO, filemode='w')  
  
  
def word\_count():  
 # For this exercise, write a program that serially counts the number of occurrences  
 # of each word in the book Alice in Wonderland.  
 #  
 # The text of Alice in Wonderland will be fed into your program line-by-line.  
 # Your program needs to take each line and do the following:  
 # 1) Tokenize the line into string tokens by whitespace  
 # Example: "Hello, World!" should be converted into "Hello," and "World!"  
 # (This part has been done for you.)  
 #  
 # 2) Remove all punctuation  
 # Example: "Hello," and "World!" should be converted into "Hello" and "World"  
 #  
 # 3) Make all letters lowercase  
 # Example: "Hello" and "World" should be converted to "hello" and "world"  
 #  
 # Store the the number of times that a word appears in Alice in Wonderland  
 # in the word\_counts dictionary, and then \*print\* (don't return) that dictionary  
 #  
 # In this exercise, print statements will be considered your final output. Because  
 # of this, printing a debug statement will cause the grader to break. Instead,  
 # you can use the logging module which we've configured for you.  
 #  
 # For example:  
 # logging.info("My debugging message")  
 #  
 # The logging module can be used to give you more control over your  
 # debugging or other messages than you can get by printing them. Messages  
 # logged via the logger we configured will be saved to a  
 # file. If you click "Test Run", then you will see the contents of that file  
 # once your program has finished running.  
 #  
 # The logging module also has other capabilities; see  
 # https://docs.python.org/2/library/logging.html  
 # for more information.  
  
 word\_counts = {}  
 exclude = set(string.punctuation)  
  
 for line in sys.stdin:  
 data = line.strip().split(" ")  
 for word in data:  
 word\_l = word.lower()  
 word\_l = ''.join(ch for ch in word\_l if ch not in exclude)  
 logging.info(word\_l)  
 if word\_l in word\_counts:  
 logging.info(word.lower())  
 word\_counts[word\_l] += 1  
 else:  
 word\_counts[word\_l] = 1  
 logging.info( word\_counts[word\_l])  
 print(word\_counts)  
  
  
word\_count()  
  
  
  
Week Six � Question #4  
import sys  
import string  
import logging  
  
logging.basicConfig(filename="logger.log", format='%(message)s',  
 level=logging.INFO, filemode='w')  
  
  
def mapper():  
 # Also make sure to fill out the reducer code before clicking "Test Run" or "Submit".  
  
 # Each line will be a comma-separated list of values. The  
 # header row WILL be included. Tokenize each row using the  
 # commas, and emit (i.e. print) a key-value pair containing the  
 # district (not state) and Aadhaar generated, separated by a tab.  
 # Skip rows without the correct number of tokens and also skip  
 # the header row.  
  
 # You can see a copy of the the input Aadhaar data  
 # in the link below:  
 # https://www.dropbox.com/s/vn8t4uulbsfmalo/aadhaar\_data.csv  
  
 # Since you are printing the output of your program, printing a debug  
 # statement will interfere with the operation of the grader. Instead,  
 # use the logging module, which we've configured to log to a file printed  
 # when you click "Test Run". For example:  
 # logging.info("My debugging message")  
 #  
 # Note that, unlike print, logging.info will take only a single argument.  
 # So logging.info("my message") will work, but logging.info("my","message") will not.  
  
 header = True  
 colnum = 12  
 col\_district = 3 # col 4 District  
 col\_addhaargen = 8 # col 9 Aadhaar generated  
 for line in sys.stdin:  
 if header:  
 header = False  
 continue  
  
 data = line.strip().split(",")  
  
 if len(data) != colnum:  
 continue  
  
 result = '{0}\t{1}'.format(data[col\_district], data[col\_addhaargen])  
 print(result)  
  
  
mapper()