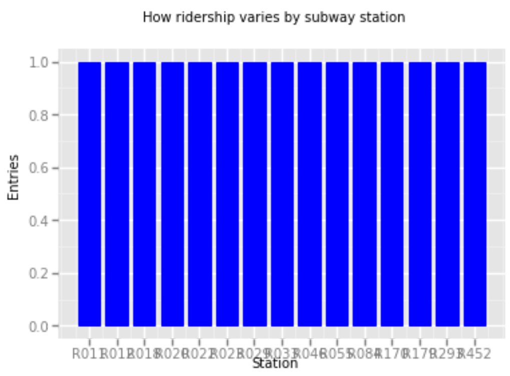
In [14]:

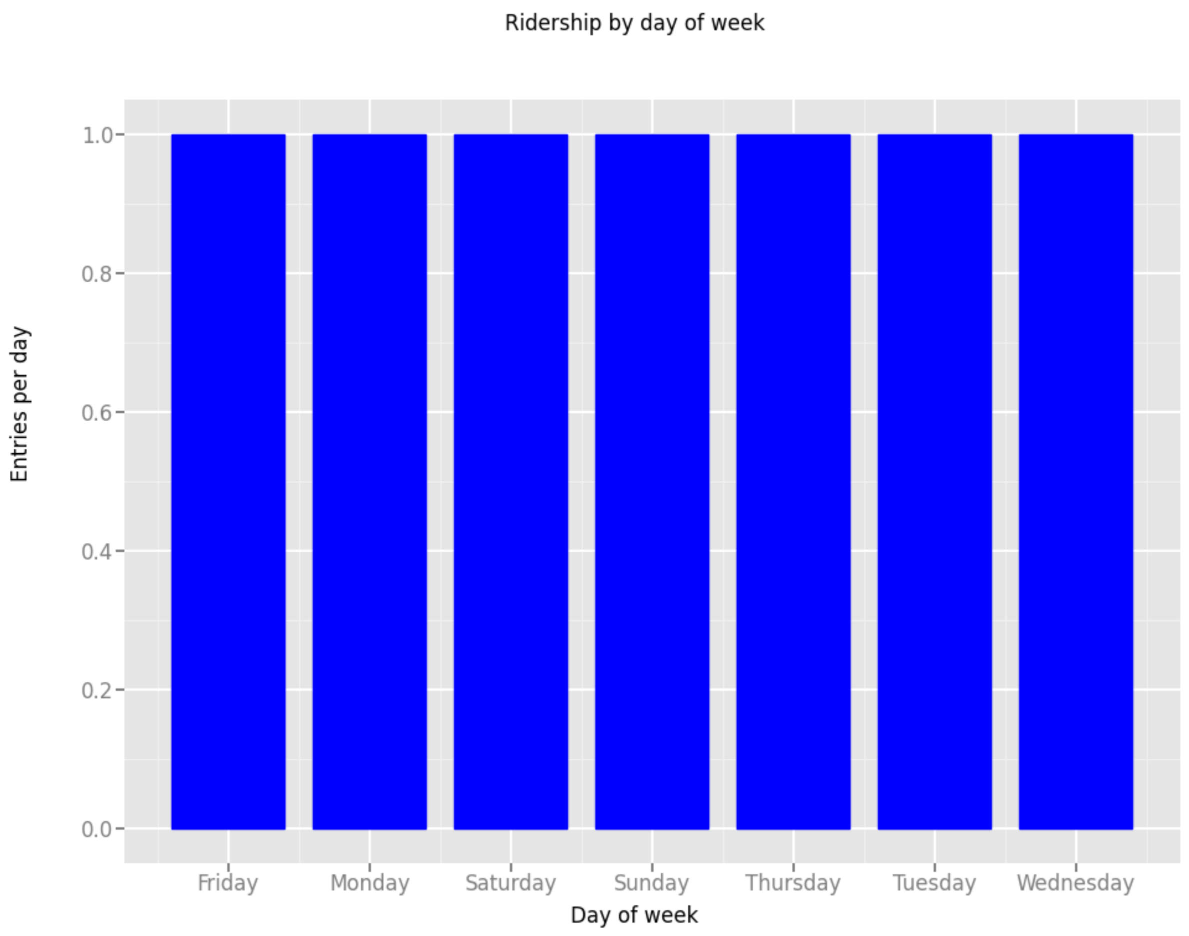
```
from pandas import *
from ggplot import *

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') 
    return plot
if __name__ == "__main__":
    filename = 'turnstile_data_master_with_weather.csv'
    df=pandas.read_csv(filename)
    output=plot_weather_data(df)
    print(output)
```



```
<ggplot: (-9223371853564030376)>
```

In [16]:

```python
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') +


    return plot

if __name__ == "__main__":
    filename = 'turnstile_data_master_with_weather.csv'
    df=pandas.read_csv(filename)
    output=plot_weather_data(df)
    print(output)
```


Ridership by day of week
Entries per day
1.0
0.8
0.6
0.4
0.2
0.0
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Day of week

```
<ggplot: (-9223371853564297740)>
```

In [1]:

```python
import logging
import sys
import string

from util import logfile #Provide util file

logging.basicConfig(filename=logfile, 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 = {}


    for line in sys.stdin:
        data = line.strip().split(" ")

        for i in data:
            key = i.translate(string.maketrans("",""),string.punctuation).lower()
            if key in word_counts.keys():
                word_counts[key] += 1
            else:
                word_counts[key] = 1

    print(word_counts)
word_count()
```

```
{}
```