Hw 1. Please complete the following activity and submit the notebook in LMS (learning management system). This homework is due by the end of week 3, to allow time for selecting your team and topic of the final project.

Q1(.5 point) Why we need Python virtual environments?

Please write no more than 3 lines.

A python virtual environment can be used to ensure you have the same exact library versions a shared project was built with. Any changes with a virtual environment will also not affect the base python package installed on your personal machine, so you can mess around in a virtual environment and keep the main python installation stable.

Q2(.5 point) Sort the following list in place alphabetically using last three characters.

In [1]:
```
continentsL = ['Europe', 'Africa', 'America', 'Antarctica', 'Asia', 'Austral
# Write you code in the next line
continentsS = sorted(continentsL, key=lambda continent: continent[-3:])
print(continentsS)
```

```
['Africa', 'America', 'Antarctica', 'Australia', 'Europe', 'Asia']
```

Q3(1= .5+.5 points) We want to compare random integer generation time in numpy and random module of "plain" python. Generate 100 random integers in the range 1 to 10 including 1 and 10. Write jupyter notebook line magic to compare execution time of random integer generation in the following cells (look for the cells having the

comment *write line magic*). Line magic must execute the given statement 10 times in a loop and repeat the process 5 times.

Hints:

- Note that numpy.random has randint function and random has random.randint function.
- Press shift+tab after typing above function or search their documentation to understand their parameters.
- use ? on magic command timeit for further documentation. see how to use **-n and -r options**.

In [2]:
```
import random
num_sample = 100
low = 1
high = 10
```

In [3]:
```
# write line magic for random module in the next line
%timeit -n10 -r5 for _ in range(num_sample): random.randint(low, high)
```

```
98 µs ± 9.46 µs per loop (mean ± std. dev. of 5 runs, 10 loops each)
```

In [4]:
```
import numpy as np
```

In [5]:
```
# write line magic for numpy randint in the next line
%timeit -n10 -r5 for _ in range(num_sample): np.random.randint(low, high + 1
```

```
283 µs ± 61.1 µs per loop (mean ± std. dev. of 5 runs, 10 loops each)
```

To start working toward final project, complete the following questions by writing markdown in the same cell in next line. Each question in worth .5 point.

In tools 1 course, ***we are concerned with data cleaning, feature engineering and exploratory analysis. Make sure your selected final project allows for lots of data cleaning, transformation and engineering opportunity.***

Go over project and presentation rubric in course webpage to better understand the requirement for the project. We will evaluate the answer to see if it meets the final project requirements.

If you are working in a team, we'll read only one team member notebook for the following questions. Please make sure you have identical answers for following questions.

To edit a markdown cell you need to double click in the cell

Q4 List your team members in bullet format. (No more than 3 members).

Note that

- **Each member** has to present a portion of the project in the final presentation.
- Final report must list the contribution of each member.

- Katie Chen
- Zachary Francis

Q5 What problem you are trying to solve?

Denver's Open Data Catalogue keeps an open record of traffic accidents from the past few years in their Open Data Catalogue. This data set can be used to possibly answer several inquiries:

- Are traffic accidents in Denver on an upward or downward trend?
- Are serious injuries or fatalites changing in any significant way?
- Are fatalaties or injuries more likely when a certain type of vehicle is at fault? Is the struck vehicle?

Other questions may arrise in exploratory data analysis. This data set is only for traffic accidents in Denver. It may be combined or compared against similar data sets for other major cities to characterize vehicle safetly in Denver against similar metropolitan areas.

Q6 Describe the attributes of examples in the dataset. If you are planning to collect the

data (webscrapping, web API etc.), list the timeline to finish the data collection.

The data set can be accessed by downloading a CSV or calling against the REST API. Ultimately it is a large file of comma separated values, with 48 columnns. Not all the columns will be of use - there is especially a lot of location/spatial data that will probably not be prudent to the goals of this data analysis. The CSV file has been downloaded and is included in the git repository of this notebook. To make comparisons with traffic data from other cities, similar data sets will need to downloaded (or to save on reposity space - access through a web api via a `curl` call). Comparing this data set against Denver Traffic Counts might also be useful to normalize the data against traffic density.

In [6]:
```
# Takes 2+ minutes to download when the curl is executed
!curl -o traffic_accidents.csv https://www.denvergov.org/media/gis/DataCatal
!head -n1 traffic_accidents.csv | tr ',' '\n' | nl
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
nt
                                 Dload  Upload   Total   Spent    Left  Speed
100  122M  100  122M    0     0   451k      0  0:04:37  0:04:37 --:--:--  221
k      0  0:11:49  0:00:06  0:11:43  198k  0   386k      0  0:05:23  0:00:19
0:05:04  699k 371k      0  0:05:36  0:00:22  0:05:14  355k 0   465k      0
0:04:28  0:00:57  0:03:31  269k  462k      0  0:04:30  0:01:19  0:03:11  507k
    0  0:04:36  0:01:25  0:03:11  293k 0:04:34  0:01:47  0:02:47  457k     0
0:04:35  0:02:02  0:02:33  437k0  0:04:42  0:02:26  0:02:16  275k  0   438k
    0  0:04:45  0:02:31  0:02:14  309k:04:46  0:02:34  0:02:12  382kk      0
0:04:44  0:02:35  0:02:09  492k 443k      0  0:04:41  0:03:00  0:01:41  365kk
      0  0:04:44  0:03:20  0:01:24  261k  448k      0  0:04:39  0:03:24  0:0
1:15  795k      0  0:04:33  0:03:53  0:00:40  458k  0   455k      0  0:04:34
0:04:31  0:00:03  464k
     1  shape
     2  object_id
     3  incident_id
     4  offense_id
     5  offense_code
     6  offense_code_extension
     7  top_traffic_accident_offense
     8  first_occurrence_date
     9  last_occurrence_date
    10  reported_date
    11  incident_address
    12  geo_x
    13  geo_y
    14  geo_lon
    15  geo_lat
    16  district_id
    17  precinct_id
    18  neighborhood_id
    19  bicycle_ind
    20  pedestrian_ind
    21  HARMFUL_EVENT_SEQ_1
    22  HARMFUL_EVENT_SEQ_2
    23  HARMFUL_EVENT_SEQ_MOST
    24  road_location
    25  ROAD_DESCRIPTION
    26  ROAD_CONTOUR
    27  ROAD_CONDITION
    28  LIGHT_CONDITION
    29  TU1_VEHICLE_TYPE
    30  TU1_TRAVEL_DIRECTION
    31  TU1_VEHICLE_MOVEMENT
    32  TU1_DRIVER_ACTION
    33  TU1_DRIVER_HUMANCONTRIBFACTOR
    34  TU1_PEDESTRIAN_ACTION
    35  TU2_VEHICLE_TYPE
    36  TU2_TRAVEL_DIRECTION
    37  TU2_VEHICLE_MOVEMENT
    38  TU2_DRIVER_ACTION
    39  TU2_DRIVER_HUMANCONTRIBFACTOR
    40  TU2_PEDESTRIAN_ACTION
    41  SERIOUSLY_INJURED
    42  FATALITIES
```

```
43  FATALITY_MODE_1
44  FATALITY_MODE_2
45  SERIOUSLY_INJURED_MODE_1
46  SERIOUSLY_INJURED_MODE_2
47  POINT_X
48  POINT_Y
```

Q7 Give example of records (sample points). Explain why this is a noisy data set (highlight portion of record attributes you think are noisy).

Some columns of interest will be the date, geospatial data if a an assessment is to be made on what driving locations or the most dangerous in Denver, the road description, the vehicle types, fatalities, and injuries. Many of the records show signs of being hand entered, and there are cases of missing data.

In [7]:
```
# Traffic accident offense
!cat traffic_accidents.csv | cut -d',' -f7 | sort | uniq -c | sort -nr | hea
```

```
151027 TRAF - ACCIDENT
64844 TRAF - ACCIDENT - HIT & RUN
6247 TRAF - ACCIDENT - DUI/DUID
3416 TRAF - ACCIDENT - SBI
2165 TRAF - ACCIDENT - POLICE
 606 TRAF - ACCIDENT - FATAL
   1 top_traffic_accident_offense
```

In [8]:
```
# Top 5 Vehicle types - passenger car and van is duplicated because of diffe
!cat traffic_accidents.csv | cut -d',' -f29 | sort | uniq -c | sort -nr | he
```

```
96848 PASSENGER CAR/VAN
48526 SUV
21640 PICKUP TRUCK/UTILITY VAN
14704 HIT AND RUN UNKNOWN
11077 Passenger Car/Passenger Van
```

In [9]:
```
# Converting case did not help, because the entries are slightly different
# Is there a distinction beween van and passenger van?
!cat traffic_accidents.csv | cut -d',' -f29 | tr [a-z] [A-Z] | sort | uniq -
```

```
96848 PASSENGER CAR/VAN
48526 SUV
24998 PICKUP TRUCK/UTILITY VAN
14704 HIT AND RUN UNKNOWN
11077 PASSENGER CAR/PASSENGER VAN
```

In [10]:
```
# Actions of the driver at fault - how to deal with other if this column is
!cat traffic_accidents.csv | cut -d',' -f32 | tr [a-z] [A-Z] | sort | uniq -
```

```
47520 CARELESS DRIVING
33414 FOLLOWED TOO CLOSELY
22304 FAILED TO YIELD ROW
22237 OTHER
16882 LANE VIOLATION
```

Q8 What kind of feature engineering opportunity you see.

We'll cover feature engineering but here are some examples to get an idea of feature engineering.

- Do you need to extract a number, year, or a word in column (attribute of a record).
  - Perhaps extracting Miss, Mr., Dr. is more important than actual name?
  - Maybe year or time is embbedded in the text desciption and it is important for analysis.
- If a column has numerical values do you see a potential for using these values or a transformation of these values like square, log etc?

Dates will need to be converted to datetime objects in order to do a proper time series analysis, or easily aggregate by month or year.

In [11]:
```
!head traffic_accidents.csv | cut -d',' -f8-10
```

```
first_occurrence_date,last_occurrence_date,reported_date
2020-01-06 14:00:00,2020-01-06 14:00:00,2020-01-07 05:46:00
2020-04-23 20:44:00,2020-04-23 20:44:00,2020-04-23 22:35:00
2020-01-21 12:40:00.000001,2020-01-21 12:40:00.000001,2020-01-21 12:40:00.000
001
2020-01-14 08:35:00,2020-01-14 08:35:00,2020-01-14 08:35:00
2020-01-18 17:30:00,2020-01-18 17:30:00,2020-01-18 17:30:00
2020-06-30 08:06:00,2020-06-30 08:06:00,2020-06-30 08:06:00
2020-02-08 20:30:00,2020-02-08 20:30:00,2020-02-12 13:01:00
2020-05-06 11:46:00,2020-05-06 11:46:00,2020-05-06 11:46:00
2020-02-24 10:30:00,2020-02-24 10:30:00,2020-02-24 10:30:00
```

The first few columns contain special ids that don't have any human meaning and will have to be decoded for them to mean anything useful

In [12]:
```
!head traffic_accidents.csv | cut -d',' -f1-6
```

shape,object_id,incident_id,offense_id,offense_code,offense_code_extension
<geoprocessing describe geometry object object at 0x036D5200>,6765,202012333,20201233354010,5401,0
<geoprocessing describe geometry object object at 0x036DC120>,6766,2020247273,202024727354010,5401,0
<geoprocessing describe geometry object object at 0x036DC120>,6767,202046084,20204608454410,5441,0
<geoprocessing describe geometry object object at 0x036E2D00>,6768,202029570,20202957054410,5441,0
<geoprocessing describe geometry object object at 0x036D5200>,6769,202040402,20204040254410,5441,0
<geoprocessing describe geometry object object at 0x036DCC60>,6770,2020397087,202039708754410,5441,0
<geoprocessing describe geometry object object at 0x036DCC60>,6771,202093434,20209343454410,5441,0
<geoprocessing describe geometry object object at 0x036E2220>,6772,2020277702,202027770254410,5441,0
<geoprocessing describe geometry object object at 0x036D53A0>,6773,2020118955,202011895554410,5441,0