DATA1002

Project 1 Report

Vehicular Crash Data Analysis

This report is for the cleaning of data as well as low level analysis/aggregate summaries of data

Minh Hoang Minh Le, Keenan Lee, Ezio Zhang, Elbert Wan

480133780, 480382908, , 470455098, 450199682

Contents

Domain Knowledge 3
Stakeholders 4
Data Sources and Accessibility 5
Datasets 6
Limitation and Strength of the Datasets 7
Cleaning / Breakdown of Code 8
  Altering data for consistency 9
    Alcohol 9
    Months 10
    Days of the week 10
    Road Conditions 11
    Weather Conditions 11
    Day and Night 12
    Collisions 13
    Severity of Incident 13
    Drug use 14
  Checks for Incorrect Values 14
    Alcohol 14
    Count 14
    Total Units 14
    Total Fatalities 14
    Injuries 15
    Crash Times 15
    Day 16
    Time 16
    Speed Limit 16
    Road Moisture 16
    Weather 16
    Illumination 17
    Collision 17
    Severity 17
    Drugs 17
  Sorting 17
Analysis 18
  Mean Total units for each nation corresponding to weather conditions 18
    Aim 18

Process 18
Results..18
Analysis of Results 18
Accidents daily in total and between weekdays and weekends of US and AUS.................19
Aim 19
Process 19
Results..19
Analysis of Results 19
Reference: 20
Appendix...21
Cleaning Code (complete) 21
Analysis...32
Daily..32
Analysis(aggregate) 33
Analysis(Daily) 37

Domain Knowledge

Vehicle accidents are one of the most common accidents in the world, and almost every single day, the news have a segment about them. [1] Road fatalities and injuries caused by the lack of attention on the road may raise some concern about the high number of cars in towns. That is why many activities and events supporting walking and public transport (e.g. the World Car Free Day) can be popular for citizens and even for the governments. For example, Australian Government announced franchising to fund into the improvement of public transport. [2]

Therefore, in two different datasets (from the US and from AUS), we analysed in what circumstances these accidents occurred, how the surrounding affected those accidents and how the surrounding is affected by those accidents.

Stakeholders

This dataset was created with the purpose of identifying and analysing the major factors that contribute towards vehicle accidents. As such, the general public and the government are the primary stakeholders for this project.

For the public, the results and observations of this dataset can allow the public to make choices that will minimise their chances of crashing. For example, if a person sees that a large majority of crashes occur at night, they can practice greater caution while driving at night.

For the government, this data will assist them in determining what factors result in the most accidents and implement policy that minimises the danger of this particular factor. This could involve imposing heavier fines during holiday periods if the dataset suggests a link between holidays and accidents or enforcing stricter speed limits for areas prone with accidents.

Data Sources and Accessibility

The South Australian dataset (2017_DATA_SA_Crash.csv) is accessible via the South Australian Government Data Directory. The file is published by the Department of Planning, Transport and Infrastructure. “This data is available for everyone” - claims the South Australian Government in order to create impact for their community and economy (cited from https://data.sa.gov.au/strategies). Source: https://data.sa.gov.au/data/dataset/road-crash-data

The publishers of our US dataset (bf8b3c7e-8d60-40df-9134-21606a451c1a) are Allegheny County (Pennsylvania) and the City of Pittsburgh. The file is accessible through the U.S. Government’s open data website where this particular file “is intended for public access and use” (cited from https://catalog.data.gov/dataset/allegheny-county-crash-data). Source: https://catalog.data.gov/dataset/allegheny-county-crash-data

Datasets

The zip file contains two csv files about vehicle accidents in certain US and Australian area. These datasets have some common columns (e.g. number of injuries, weather condition, road moisture) and many differences (for example the US dataset is more detailed by having more columns or wider range of categorical values within a column). The zip file also contains information related to the metadata and dictionary of the US dataset, and a dictionary is provided about the Australian dataset.

All the columns of our combined and transformed dataset have the same meaning and correspond to the columns of the original dataset with the exception of one. For the column ‘Adverse Weather Conditions’ in our combined dataset, we changed its value to either “Y”, signifying that there were adverse weather conditions and “N”, signifying that there were no adverse weather conditions. In the original two datasets the columns contained information on the weather such as “raining” or “hail”, so we merely transformed these values to either “Y” or “N”.

Limitation and Strength of the Datasets

The strengths of our final dataset is that it contains many rows and columns. The large number of rows provides increases the reliability of any conclusions drawn from the data due to the large sample size. The large number of columns allows the user to investigate the relationship between many variables and ensures that most confounding variables can be accounted for. These columns also contain a wide range of detailed values which provides greater accuracy when drawing conclusions.

The two datasets which were combined also come from Western nations with similar standards of living. This minimises any misleading conclusions that could have been obtained from combining datasets from two nations with very different cultures and standards of living.

The limitations of our final dataset is that in combining the original datasets, not all information could be retained. Columns that could not be linked between the original datasets had to be removed. Further, certain columns for one dataset contained more information compared to the other dataset and as a result information was lost. The US and Australia also have different units of speed (mp/h compared to km/h). The combined dataset converted the speed limits from mp/h to km/h, however as these converted values were not identical the to speed limits of the Australian dataset, this column may produce misleading results.

Cleaning / Breakdown of Code

By having both csv files and the cleaning.py file in the same directory, a modified final.csv file can be created, and will be used to do some analysis later on. In this section of the report, each code segment of the cleaning.py is going to be elaborated on. Additionally, more details are implemented into the python file as comment.

The program first opens the 2017_DATA_SA_Crash.csv file, and by using for loop, we are able to work with each row (item) of the dataset (356-414). Within this loop we chose 16 attributes (columns, represented as line[number] in the code) and added a "Nation" attributes to distinguish the items of the two datasets. In the end, each modified row (line 398) has to go through the *cleaningFunction* (line 402) which return a Boolean and a list. If the row has any incorrect, default, inconsistent or missing value, the Boolean value will be false, and the row will not be written in our final.csv file (line 404-414). The list contains the value of each attributes. from line 417 to 471, the same logic is applied to the US f8b3c7e-8d60-40df-9134-21606a451c1a.csv dataset, except since this dataset uses mp/h in their speed limit column, we converted to km/h and rounded the number.

cleaningFunction (line 13-354)

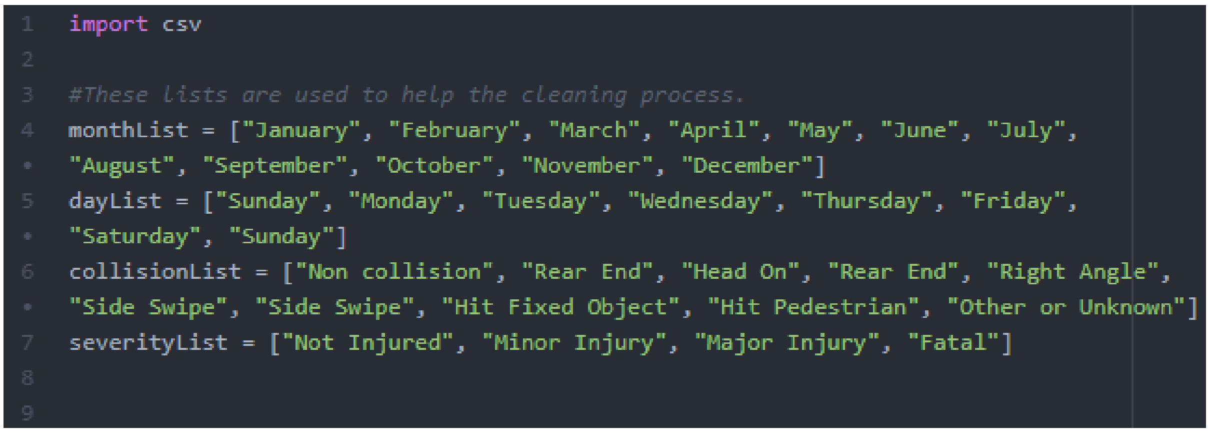
In this function, missing, default, inconsistent and incorrect values were detected. As soon as one of them appeared in the row, a false Boolean value is returned. 16 attributes were chosen based on how similarly those attributes works. For instance, both files have column showing the number of minor injuries and they both only contain numerical type of values. In this case, we only searched for missing, default, inconsistent and incorrect values (empty cell, negative number, float numbers etc.).

However, there were some attributes where the values were represented differently: for example, in the US dataset whether the road was wet or dry were represented as numbers (dry=0, wet=1). Furthermore, there were attributes where the range of categorical values differed in the US dataset from the AUS dataset. In the column crash type, for example, even though some of the categories were the same (both datasets have "Rear End" category in their column crash/collision type), the US attributes were more detailed (in the US collision type, there are 10 different categories while in the AUS, it is way less).

Therefore, we adjusted the number of different text values based on the AUS dataset (for example we did not included items from the US file where the collision type is "Rear-to-rear" since the AUS one does not have this option).

After adjusting the values of each columns to our preference, cleaning was done accordingly. For more details, please open the cleaning.py file and read the comments.

Below is the broken down code that we used for cleaning:
MonthList, DayList, CollisionList, SeverityList were created for the purpose of confirming accurate data in the dataset



```
import csv

#These lists are used to help the cleaning process.
monthList = ["January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"]
dayList = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"]
collisionList = ["Non collision", "Rear End", "Head On", "Rear End", "Right Angle",
"Side Swipe", "Side Swipe", "Hit Fixed Object", "Hit Pedestrian", "Other or Unknown"]
severityList = ["Not Injured", "Minor Injury", "Major Injury", "Fatal"]
```

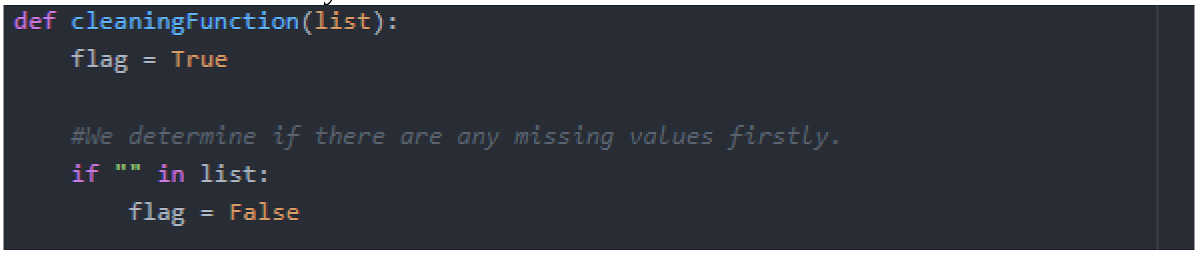
This function was created to clean the data provided by checking each row of data for

- *Incorrect or inconsistent data values*
- *Missing or obsolete data values*

Altering data for consistency

Notes to be made on this include:

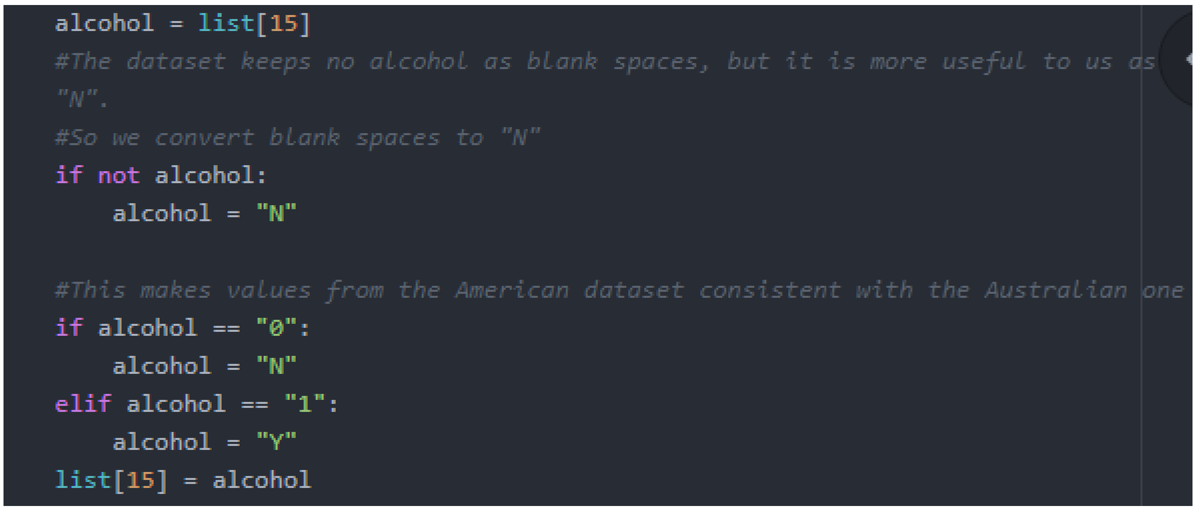
- *A flag is used to detect if any values are missing, if they are, row is removed as it is unable to be used in analysis*



```
def cleaningFunction(list):
    flag = True

    #We determine if there are any missing values firstly.
    if "" in list:
        flag = False
```

Alcohol

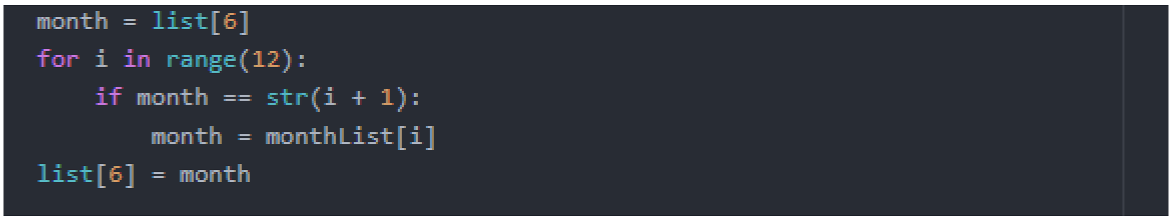
- *US dataset uses binary (1 for yes, 0 for no) while Australian dataset uses Y/N, to make data consistent, US data was changed from '1' to 'Y', and '0' to 'N'*
- *Australian dataset uses ~~'blank'~~ for 'no DUI', thus it was changed to N for 'no DUI'*



```
    alcohol = list[15]
    #The dataset keeps no alcohol as blank spaces, but it is more useful to us as
    "N".
    #So we convert blank spaces to "N"
    if not alcohol:
        alcohol = "N"

    #This makes values from the American dataset consistent with the Australian one
    if alcohol == "0":
        alcohol = "N"
    elif alcohol == "1":
        alcohol = "Y"
    list[15] = alcohol
```

Months

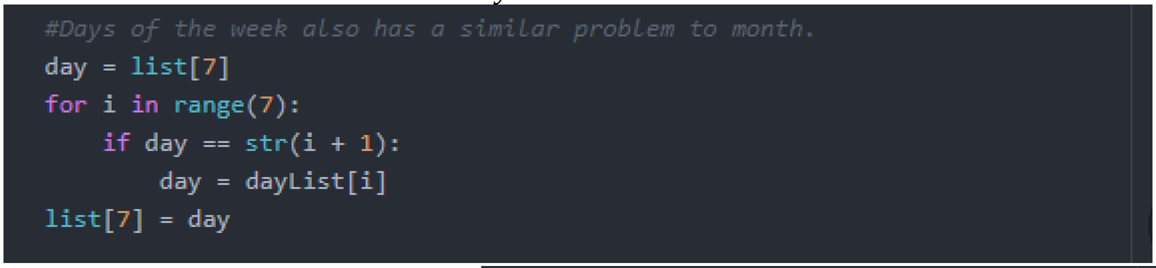
- *US dataset uses numerical value for months (1 for January, 2 for February etc.)*
- *Australian dataset uses names for months (January for January, February for February)*
- *To make it easier to see, we decided to use names instead of numbers, thus the US dataset was altered to have Month names*



```
month = list[6]
for i in range(12):
    if month == str(i + 1):
        month = monthList[i]
list[6] = month
```

Days of the week

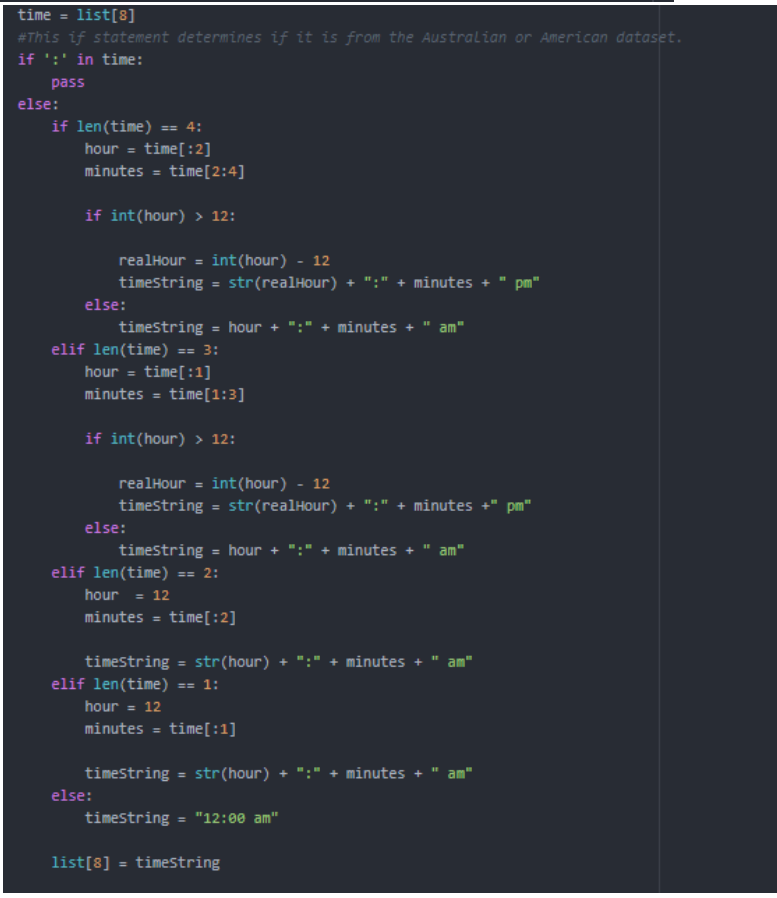
- *US dataset uses numerical value for days (1 for Sunday, 2 for Monday etc.)*
- *Australian dataset uses names for days (Monday for Monday, Tuesday for Tuesday etc.)*
- *To make it easier to see, we decided to use names instead of numbers, thus the US dataset was altered to have Day names*



```
#Days of the week also has a similar problem to month.
day = list[7]
for i in range(7):
    if day == str(i + 1):
        day = dayList[i]
list[7] = day
```

Time

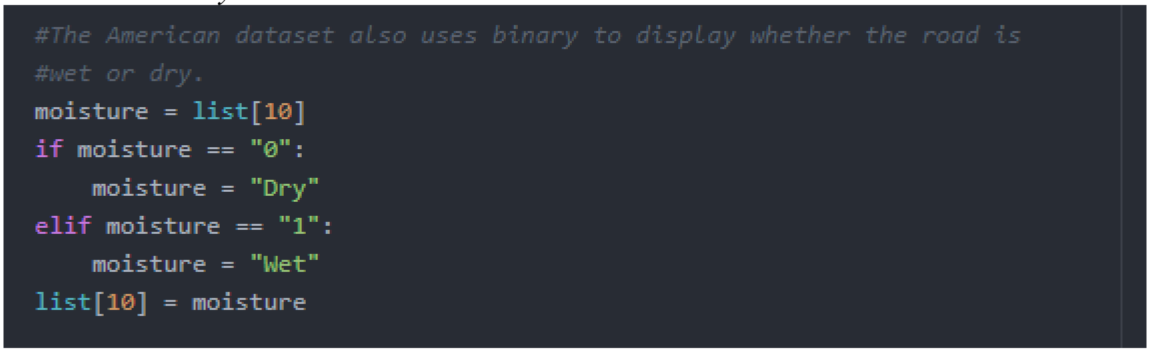
- *US dataset uses 24 hour time*
- *Australian dataset uses 12 hour time*
- *To make data consistent and easier to view, US dataset was changed to 12 hour format.*



```
time = list[8]
#This if statement determines if it is from the Australian or American dataset.
if ':' in time:
    pass
else:
    if len(time) == 4:
        hour = time[:2]
        minutes = time[2:4]

        if int(hour) > 12:

            realHour = int(hour) - 12
            timeString = str(realHour) + ":" + minutes + " pm"
        else:
            timeString = hour + ":" + minutes + " am"
    elif len(time) == 3:
        hour = time[:1]
        minutes = time[1:3]

        if int(hour) > 12:

            realHour = int(hour) - 12
            timeString = str(realHour) + ":" + minutes +" pm"
        else:
            timeString = hour + ":" + minutes + " am"
    elif len(time) == 2:
        hour  = 12
        minutes = time[:2]

        timeString = str(hour) + ":" + minutes + " am"
    elif len(time) == 1:
        hour = 12
        minutes = time[:1]

        timeString = str(hour) + ":" + minutes + " am"
    else:
        timeString = "12:00 am"

    list[8] = timeString
```

Road Conditions

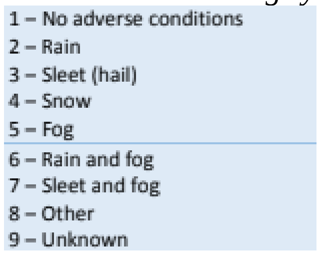
- *US dataset uses binary to determine if road was wet or dry ('0' for dry, '1' for wet*
- *Australian dataset uses 'wet' or 'dry' to classify data.*
- *To make data consistent and easier to understand, US dataset was changed to 'wet' and 'dry'.*



```
#The American dataset also uses binary to display whether the road is
#wet or dry.
moisture = list[10]
if moisture == "0":
    moisture = "Dry"
elif moisture == "1":
    moisture = "Wet"
list[10] = moisture
```

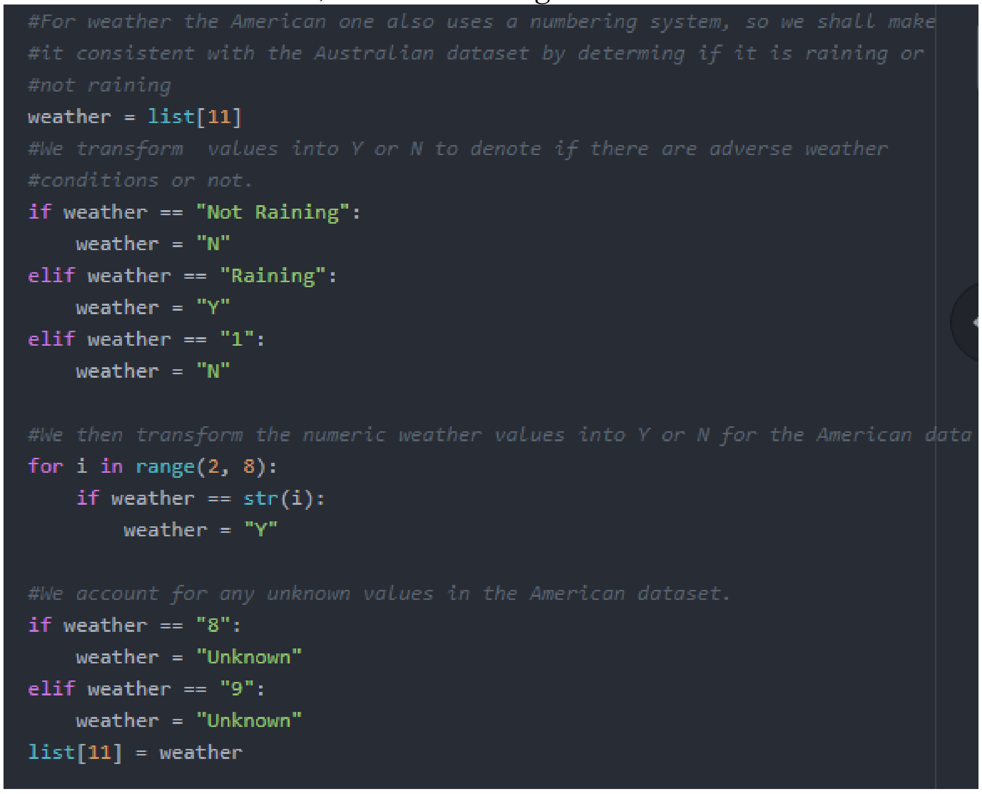
Weather Conditions

- *US dataset uses a numbering system for weather conditions*



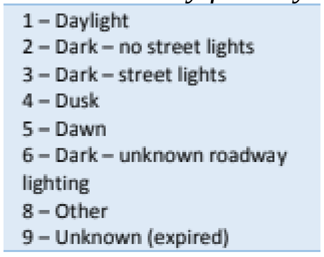
1 – No adverse conditions
2 – Rain
3 – Sleet (hail)
4 – Snow
5 – Fog
6 – Rain and fog
7 – Sleet and fog
8 – Other
9 – Unknown

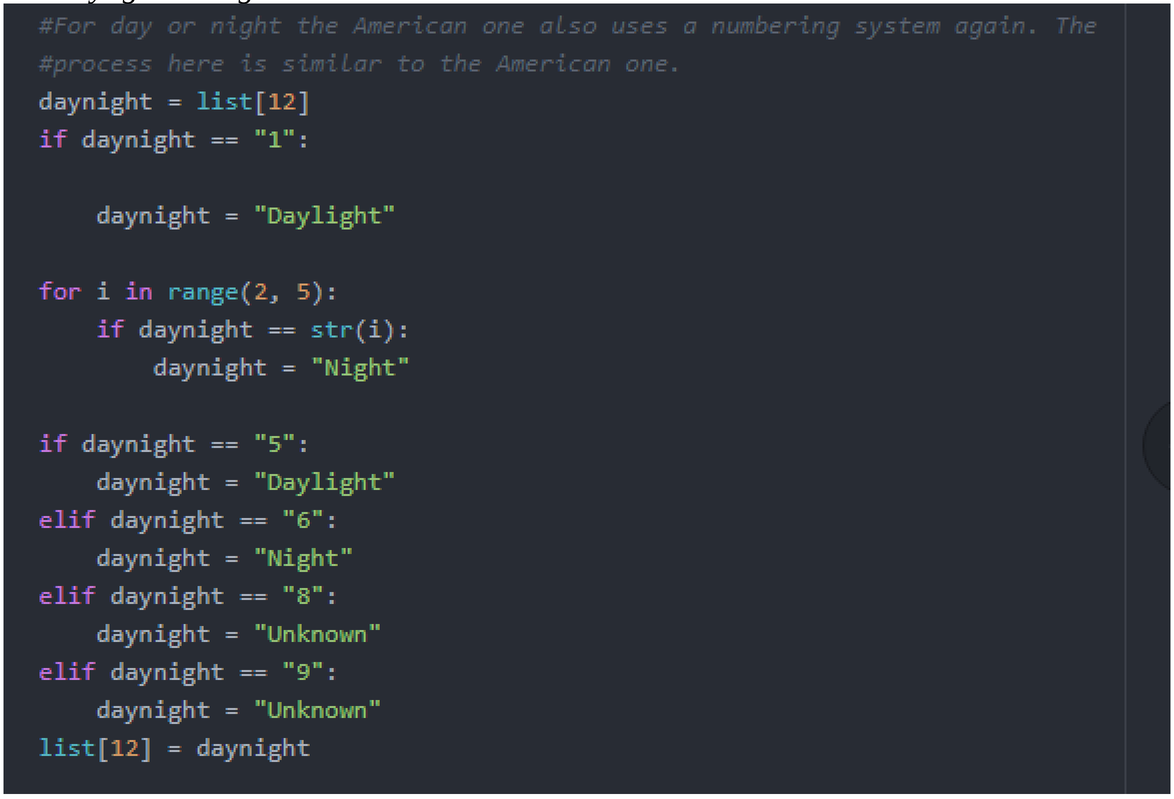
- *Australian dataset uses either 'Raining', 'Not Raining' or 'unknown' for weather conditions*
- *To make data consistent, we changed the data to "adverse weather conditions (Y/N/unknown)" with 'Y' being adverse weather conditions, 'N' being clear weather conditions, 'unknown' being unknown weather conditions*



```
#For weather the American one also uses a numbering system, so we shall make
#it consistent with the Australian dataset by determing if it is raining or
#not raining
weather = list[11]
#We transform  values into Y or N to denote if there are adverse weather
#conditions or not.
if weather == "Not Raining":
    weather = "N"
elif weather == "Raining":
    weather = "Y"
elif weather == "1":
    weather = "N"

#We then transform the numeric weather values into Y or N for the American data
for i in range(2, 8):
    if weather == str(i):
        weather = "Y"

#We account for any unknown values in the American dataset.
if weather == "8":
    weather = "Unknown"
elif weather == "9":
    weather = "Unknown"
list[11] = weather
```

Day and Night

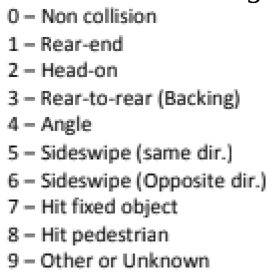
- *US dataset uses binary for Day and Night*
    - 
      1 – Daylight
      2 – Dark – no street lights
      3 – Dark – street lights
      4 – Dusk
      5 – Dawn
      6 – Dark – unknown roadway lighting
      8 – Other
      9 – Unknown (expired)
- *Australian dataset uses 'Daylight' for Day and 'Night' Night*
- *To make data consistent, we categorise the American System to either Daylight or Night.*



```
#For day or night the American one also uses a numbering system again. The
#process here is similar to the American one.
daynight = list[12]
if daynight == "1":

    daynight = "Daylight"

for i in range(2, 5):
    if daynight == str(i):
        daynight = "Night"

if daynight == "5":
    daynight = "Daylight"
elif daynight == "6":
    daynight = "Night"
elif daynight == "8":
    daynight = "Unknown"
elif daynight == "9":
    daynight = "Unknown"
list[12] = daynight
```

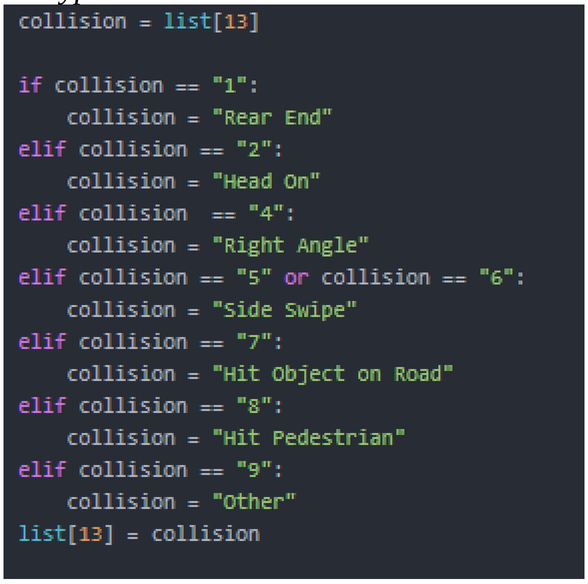
Collisions

- *US dataset used numbering system to define collision types*



0 – Non collision
1 – Rear-end
2 – Head-on
3 – Rear-to-rear (Backing)
4 – Angle
5 – Sideswipe (same dir.)
6 – Sideswipe (Opposite dir.)
7 – Hit fixed object
8 – Hit pedestrian
9 – Other or Unknown

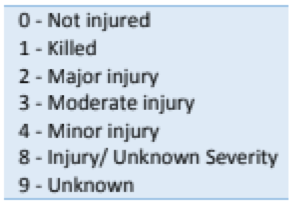
- *Australian dataset used a different classification to define collision types*
- *Code below changes the numbers in the US dataset to corresponding names. However, since the two countries different in classification, it was decided that it would not be included in the final dataset as analysis on these collision types would be inconsistant*



```
collision = list[13]

if collision == "1":
    collision = "Rear End"
elif collision == "2":
    collision = "Head On"
elif collision  == "4":
    collision = "Right Angle"
elif collision == "5" or collision == "6":
    collision = "Side Swipe"
elif collision == "7":
    collision = "Hit Object on Road"
elif collision == "8":
    collision = "Hit Pedestrian"
elif collision == "9":
    collision = "Other"
list[13] = collision
```

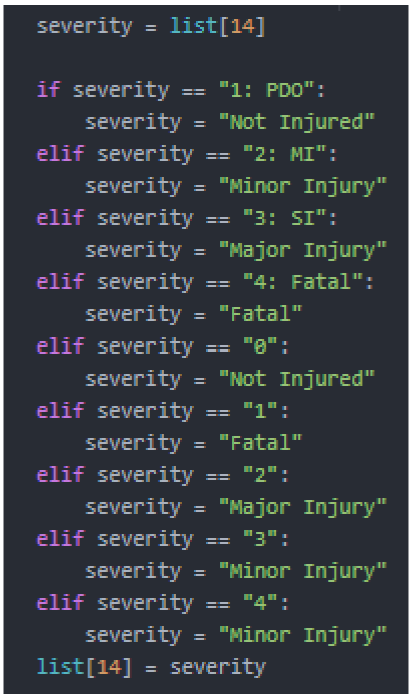
Severity of Incident

- *US dataset uses a numbering system for injuries sustained in collisions*



0 - Not injured
1 - Killed
2 - Major injury
3 - Moderate injury
4 - Minor injury
8 - Injury/ Unknown Severity
9 - Unknown

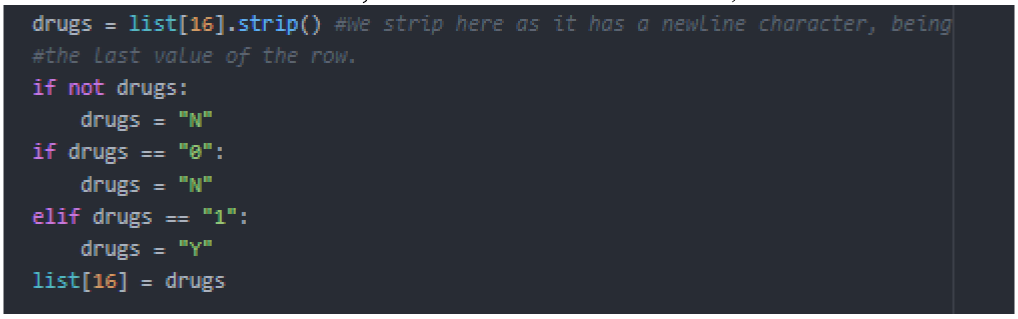
- *Australian dataset had*
  - *PDO – property damage only*
  - *MI – Medium Injury*
  - *SI – Serious Injury*
- *To make data consistent and easier to see, we classified both datasets into three severities: 'Not injured', 'Major Injury' and 'Fatal'*



```
severity = list[14]

if severity == "1: PDO":
    severity = "Not Injured"
elif severity == "2: MI":
    severity = "Minor Injury"
elif severity == "3: SI":
    severity = "Major Injury"
elif severity == "4: Fatal":
    severity = "Fatal"
elif severity == "0":
    severity = "Not Injured"
elif severity == "1":
    severity = "Fatal"
elif severity == "2":
    severity = "Major Injury"
elif severity == "3":
    severity = "Minor Injury"
elif severity == "4":
    severity = "Minor Injury"
list[14] = severity
```

Drug use

- *US dataset uses binary system to classify drug use ('1' for drugs, '0' for no drugs*
- *Australian dataset uses (Y/~~blank~~) to classify drug use ('Y' for drugs, ~~'blank'~~ for no drugs)*
- *To make data consistent, we converted ~~'blank'~~ to 'N', '1' to 'Y' and '0' to 'N'*

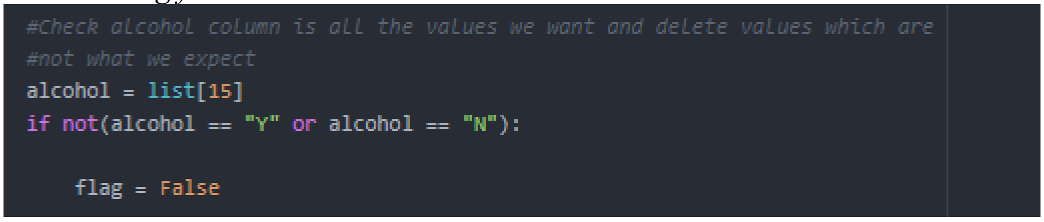


```python
drugs = list[16].strip() #We strip here as it has a newline character, being
#the last value of the row.
if not drugs:
    drugs = "N"
if drugs == "0":
    drugs = "N"
elif drugs == "1":
    drugs = "Y"
list[16] = drugs
```

Checks for Incorrect Values

Alcohol

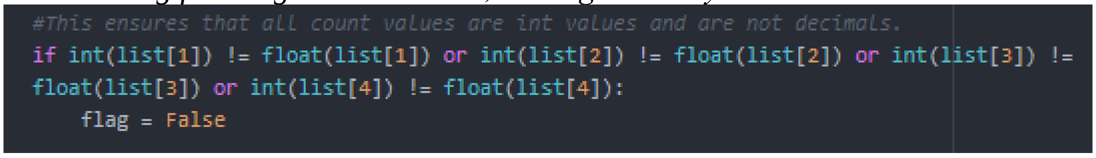
- *Checking for values that are not 'Y' or 'N'*



```python
#Check alcohol column is all the values we want and delete values which are
#not what we expect
alcohol = list[15]
if not(alcohol == "Y" or alcohol == "N"):

    flag = False
```

Count

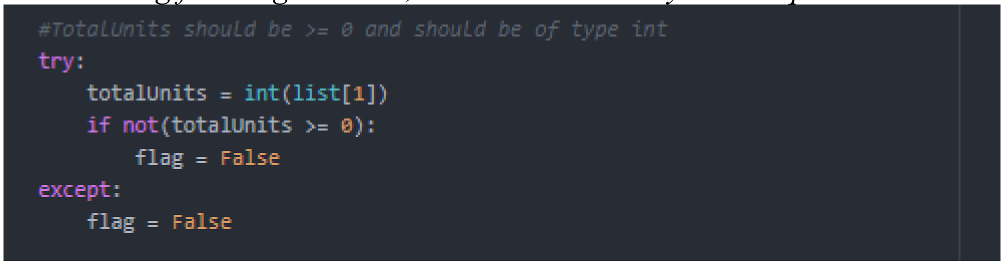
- *Checking for integer count values, making sure they aren't blank or decimals*



```python
#This ensures that all count values are int values and are not decimals.
if int(list[1]) != float(list[1]) or int(list[2]) != float(list[2]) or int(list[3]) !=
float(list[3]) or int(list[4]) != float(list[4]):
    flag = False
```

Total Units

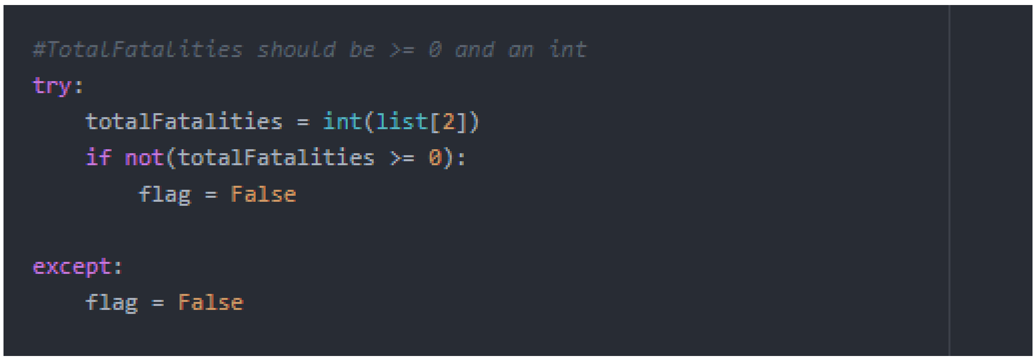
- *Checking for integer values, and makes sure they're all equal to or above 0*



```python
#TotalUnits should be >= 0 and should be of type int
try:
    totalUnits = int(list[1])
    if not(totalUnits >= 0):
        flag = False
except:
    flag = False
```

Total Fatalities

- *Checking for integer values, and makes sure they're all equal to or above 0*

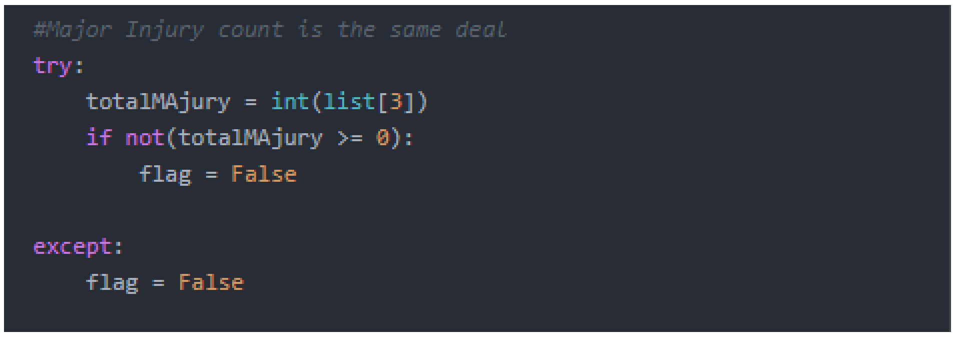


```
#TotalFatalities should be >= 0 and an int
try:
    totalFatalities = int(list[2])
    if not(totalFatalities >= 0):
        flag = False

except:
    flag = False
```

Injuries

Major Injuries

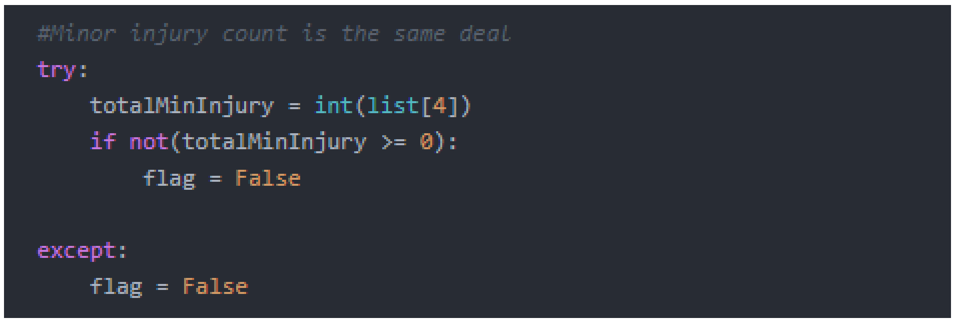
- *Checking for integer values, and makes sure they're all equal to or above 0*



```
#Major Injury count is the same deal
try:
    totalMAjury = int(list[3])
    if not(totalMAjury >= 0):
        flag = False

except:
    flag = False
```

Minor Injuries

- *Checking for integer values, and makes sure they're all equal to or above 0*

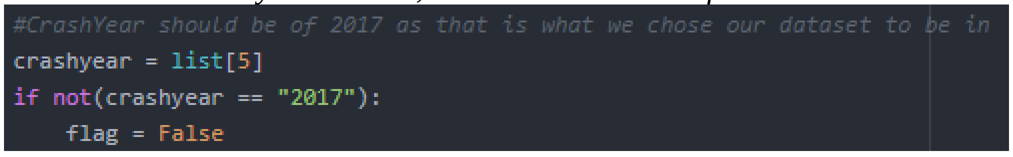


```
#Minor injury count is the same deal
try:
    totalMinInjury = int(list[4])
    if not(totalMinInjury >= 0):
        flag = False

except:
    flag = False
```

Crash Times

Crash Year

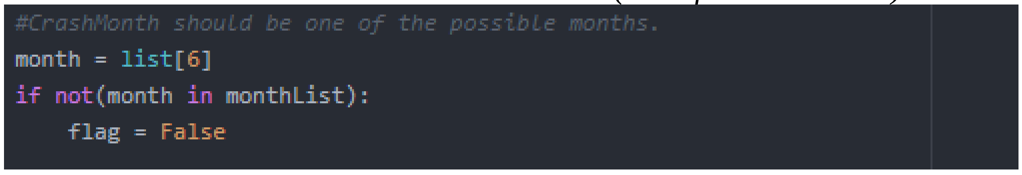
- *Checks that the year is 2017, as that is the criteria for our dataset*



```
#CrashYear should be of 2017 as that is what we chose our dataset to be in
crashyear = list[5]
if not(crashyear == "2017"):
    flag = False
```

Crash Month

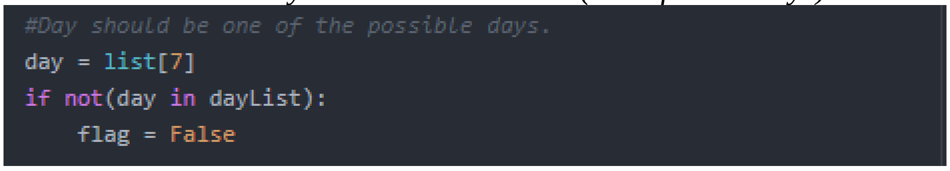
- *Checks that the months in dataset are valid (one of the 12 months)*



```
#CrashMonth should be one of the possible months.
month = list[6]
if not(month in monthList):
    flag = False
```

Day

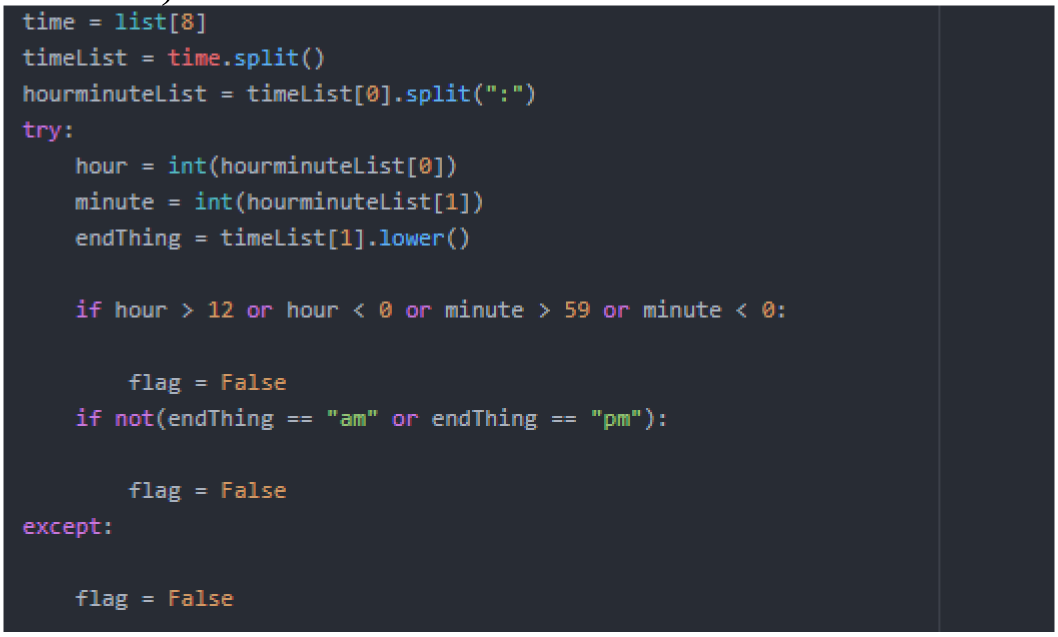
- *Checks that the days in dataset are valid (one of seven days)*



```python
#Day should be one of the possible days.
day = list[7]
if not(day in dayList):
    flag = False
```

Time

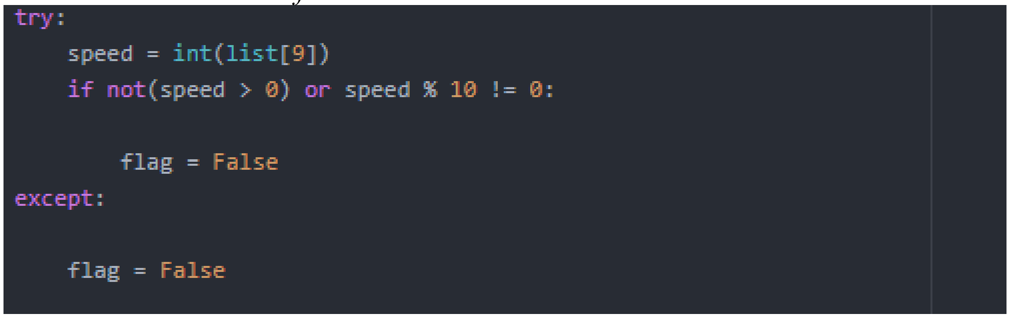
- *Checks that time is valid (hours between 0 and 12, minutes between or equal to 0 and 59)*



```python
time = list[8]
timeList = time.split()
hourminuteList = timeList[0].split(":")
try:
    hour = int(hourminuteList[0])
    minute = int(hourminuteList[1])
    endThing = timeList[1].lower()

    if hour > 12 or hour < 0 or minute > 59 or minute < 0:

        flag = False
    if not(endThing == "am" or endThing == "pm"):

        flag = False
except:

    flag = False
```

Speed Limit

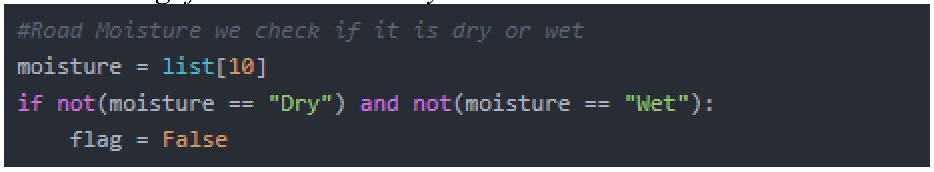
- *Checks that speed-limit is larger than 0 as well as an integer. Also checks that it's in increments of 10*



```python
try:
    speed = int(list[9])
    if not(speed > 0) or speed % 10 != 0:

        flag = False
except:

    flag = False
```

Road Moisture

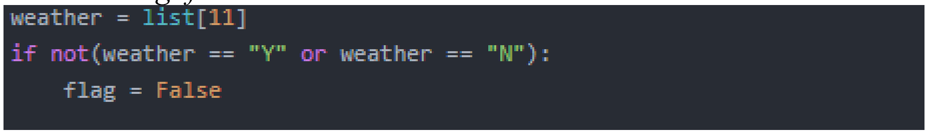
- *Checking if data is either 'Dry' or 'Wet'*



```python
#Road Moisture we check if it is dry or wet
moisture = list[10]
if not(moisture == "Dry") and not(moisture == "Wet"):
    flag = False
```

Weather

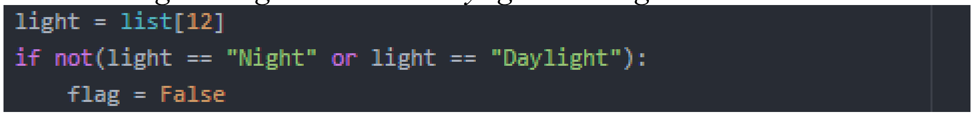
- *Checking if weather is either 'Y' or 'N'*



```python
weather = list[11]
if not(weather == "Y" or weather == "N"):
    flag = False
```

Illumination

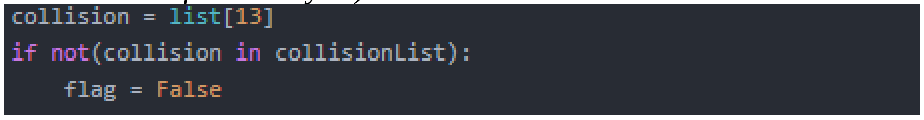
- *Checking that light is either ‘Daylight’ or ‘Night’*



```
light = list[12]
if not(light == "Night" or light == "Daylight"):
    flag = False
```

Collision

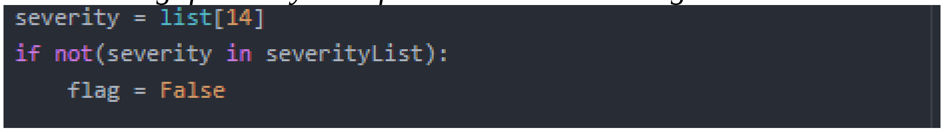
- *Checking if crash type follows correct wording (although this is unnecessary as unused in final analysis)*



```
collision = list[13]
if not(collision in collisionList):
    flag = False
```

Severity

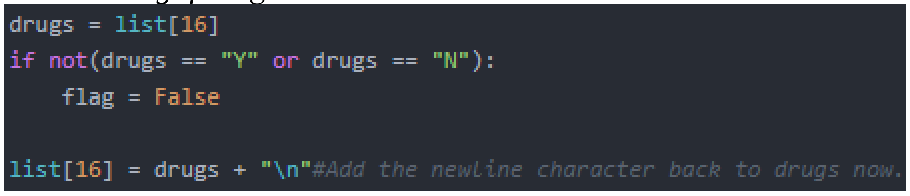
- *Checking if severity data follows correct wording*



```
severity = list[14]
if not(severity in severityList):
    flag = False
```

Drugs

- *Checking if drug data is either ‘Y’ or ‘N’*



```
drugs = list[16]
if not(drugs == "Y" or drugs == "N"):
    flag = False

list[16] = drugs + "\n"#Add the newline character back to drugs now.
```

Sorting

- *Both datasets (Australian and US) are both sorted to fit the right categories at the right columns.*
- *If it has been flagged as false, then the line is not written, if it isn’t, then the flag is written, thus this ensures that valid, consistent data is input into the ‘cleaned’ dataset*

Analysis

<u>All the files have to be in the same directory.</u>
<u>Our aggregate program can be found within our submitted files. The output of our program can be seen in a file which our program writes to for legibility (analysis.txt).</u>

Mean Total units for each nation corresponding to weather conditions

Aim

The first aggregate we chose to determine was the mean total amount of units for each nation corresponding to the two weather conditions (adverse or not adverse).

Process

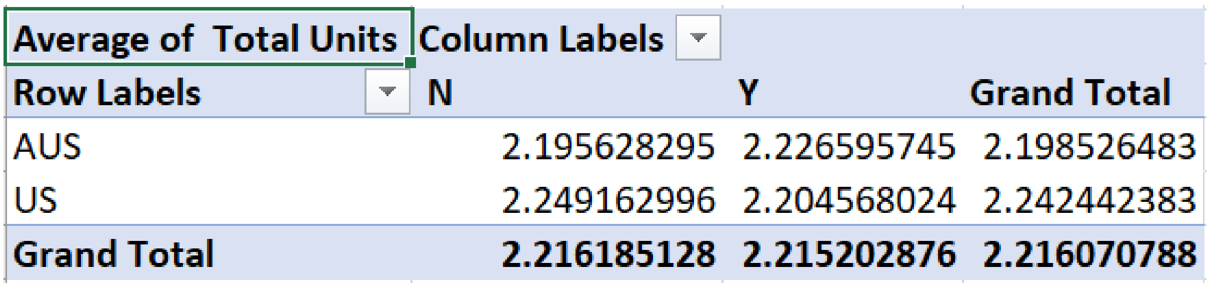
It was done through a Python program called 'aggregate.py' and we also utilized libraries such as 'statistics'. Our program creates a dictionary which groups all the combinations of nations and weather conditions together as dictionary keys. The item of each dictionary key is a list containing all the number of units satisfying the information specified in the dictionary key.

Once the lists for each nation and weather condition have been created, we then using the mean function from the statistics module to find the mean of each list. For each dictionary key, the program will then print out that dictionary key and its mean.

We then use list concatenation to combine the positive and negative weather lists for each nation. We then apply the mean function once more on this combined list to find the overall average for each nation.

Results

It is intended to replicate the pivot table:



Average of Total Units	Column Labels		
Row Labels	N	Y	Grand Total
AUS	2.195628295	2.226595745	2.198526483
US	2.249162996	2.204568024	2.242442383
Grand Total	**2.216185128**	**2.215202876**	**2.216070788**

Analysis of Results

Strangely, it seems like positive weather conditions resulted in higher amounts of people involved in crashes for the US. Overall, the US seems to have a higher average amount of people involved in crashes.

The second analysis is about the number of injuries corresponding to weather condition and counties. We observed how many people suffered injury by vehicle accidents in the USA and in Australia based on weather condition.

To calculate that, we used *pandas* library and grouped the table based on **Nation** and **Adverse Weather Conditions** columns. We extracted the number of minor and major injuries, and we also created a new table for no injuries using *np.where* function (if there was no injury it returned 1, otherwise 0). In the and we used *sum* function to receive the number and *to_dict* function to put those values into dictionaries.

According to this analysis, there were more injuries both in US and AUS when it did not rain. However, we cannot conclude that weather does not affect car crash with injury, since we do not have enough data about the weather.

Furthermore, we can assume that there are more vehicle related injuries in Australia than in the US (AUS have almost twice as many injuries then US) and that minor injuries are around ten times more common than major injuries generally.

Accidents daily in total and between weekdays and weekends of US and AUS

Aim

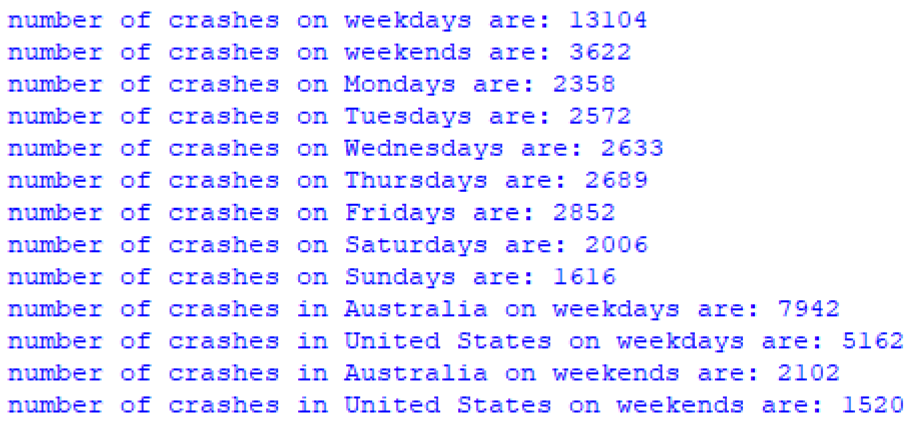
The second aggregate we chose to determine was the total amount of crashes between the days of the week as well as comparing weekdays and weekends, for total and between the two nations.

Process

This was also done through the python script "Aggregate.py". The script created a counter for Weekdays, Weekends, WeekdayAUS, WeekdayUS, WeekendAUS, WeekendUS, and all of the days of the week. It then looped through the entire csv file to total the amount of crashes for each of these categories. Finally, it prints the results on the shell.

Results

Below are the results of running the script



```
number of crashes on weekdays are: 13104
number of crashes on weekends are: 3622
number of crashes on Mondays are: 2358
number of crashes on Tuesdays are: 2572
number of crashes on Wednesdays are: 2633
number of crashes on Thursdays are: 2689
number of crashes on Fridays are: 2852
number of crashes on Saturdays are: 2006
number of crashes on Sundays are: 1616
number of crashes in Australia on weekdays are: 7942
number of crashes in United States on weekdays are: 5162
number of crashes in Australia on weekends are: 2102
number of crashes in United States on weekends are: 1520
```

Analysis of Results

It was hypothesised that there would be more crashes on the weekend because it's usually the time that people go out and party and such. However the data shows that Friday has the largest amount of crashes. This could be linked to the final day of work in the week, and a rush to get home or out to celebrate for a start to the weekend. Furthermore, the data shows the Sundays have the lowest amount of crashes. This could possibly be linked to the fact that it's a rest day before working starts again, thus most people stay at home. It can be seen that crashes rise at a proportional level starting from Sunday until Friday, and that Australia seems to have more incidents than US on both weekdays AND weekends, however this may be incorrect if data is missing or incomplete.

Reminder: these datasets are from certain areas of the US and AUS states and not entirely from US and AUS.

Reference:
[1] https://www.budgetdirect.com.au/car-insurance/research/car-accident-statistics.html
[2] http://infrastructureaustralia.gov.au/news-media/media-releases/files/Customer_Focused_Franchising.pdf

Appendix

Cleaning Code (complete)

```
import csv

#These lists are used to help the cleaning process.
monthList = ["January", "February", "March", "April", "May",
"June", "July", "August", "September", "October", "November",
"December"]
dayList = ["Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday"]
collisionList = ["Non collision", "Rear End", "Head On", "Rear
End", "Right Angle", "Side Swipe", "Side Swipe", "Hit Fixed
Object", "Hit Pedestrian", "Other or Unknown"]
severityList = ["Not Injured", "Minor Injury", "Major Injury",
"Fatal"]


#The purpose of this function is for overall cleaning. Given a
particular write
#from the dataset, it checks for missing values and accounts
for inconsistent
#and incorrect values.
def cleaningFunction(list):
    flag = True

    #We determine if there are any missing values firstly.
    if "" in list:
        flag = False

    #For alcohol there is an inconsistency as the American
dataset uses
    #binary notation while the Australian one uses Y/N. We
will convert the
    #binary to Y/N
    alcohol = list[15]
    #The dataset keeps no alcohol as blank spaces, but it is
more useful to us as "N".
    #So we convert blank spaces to "N"
    if not alcohol:
        alcohol = "N"

    #This makes values from the American dataset consistent
with the Australian one
    if alcohol == "0":
        alcohol = "N"
    elif alcohol == "1":
        alcohol = "Y"
    list[15] = alcohol
    #Inconsistency for month as the American dataset uses the 
numerical value
    #for the month instead of the name of the month like the 
Australian one.
    month = list[6]
    for i in range(12):
        if month == str(i + 1):
            month = monthList[i]
    list[6] = month

    #Days of the week also has a similar problem to month.
    day = list[7]
    for i in range(7):
        if day == str(i + 1):
            day = dayList[i]
    list[7] = day

    #Time in the American dataset is in 24 hour time instead 
of 12 hour like
    #the Australian one. We shall convert the Amercian time 
into 12 hour time
    time = list[8]
    #This if statement determines if it is from the Australian 
or American dataset.
    if ':' in time:
        pass
    else:
        if len(time) == 4:
            hour = time[:2]
            minutes = time[2:4]

            if int(hour) > 12:

                realHour = int(hour) - 12
                timeString = str(realHour) + ":" + minutes + " 
pm"
            else:
                timeString = hour + ":" + minutes + " am"
        elif len(time) == 3:
            hour = time[:1]
            minutes = time[1:3]

            if int(hour) > 12:

                realHour = int(hour) - 12
                timeString = str(realHour) + ":" + minutes +" 
pm"
            else:
                timeString = hour + ":" + minutes + " am"
        elif len(time) == 2:
            hour  = 12
            minutes = time[:2]

            timeString = str(hour) + ":" + minutes + " am"
        elif len(time) == 1:
            hour = 12
            minutes = time[:1]

            timeString = str(hour) + ":" + minutes + " am"
        else:
            timeString = "12:00 am"

        list[8] = timeString


    #The American dataset also uses binary to display whether
the road is
    #wet or dry.
    moisture = list[10]
    if moisture == "0":
        moisture = "Dry"
    elif moisture == "1":
        moisture = "Wet"
    list[10] = moisture

    #For weather the American one also uses a numbering
system, so we shall make
    #it consistent with the Australian dataset by determing if
it is raining or
    #not raining
    weather = list[11]
    #We transform  values into Y or N to denote if there are
adverse weather
    #conditions or not.
    if weather == "Not Raining":
        weather = "N"
    elif weather == "Raining":
        weather = "Y"
    elif weather == "1":
        weather = "N"

    #We then transform the numeric weather values into Y or N
for the American data
    for i in range(2, 8):
        if weather == str(i):
            weather = "Y"

    #We account for any unknown values in the American
dataset.
    if weather == "8":
        weather = "Unknown"
    elif weather == "9":
        weather = "Unknown"
    list[11] = weather

    #For day or night the American one also uses a numbering
system again. The
    #process here is similar to the American one.
    daynight = list[12]
    if daynight == "1":

        daynight = "Daylight"

    for i in range(2, 5):
        if daynight == str(i):
            daynight = "Night"

    if daynight == "5":
        daynight = "Daylight"
    elif daynight == "6":
        daynight = "Night"
    elif daynight == "8":
        daynight = "Unknown"
    elif daynight == "9":
        daynight = "Unknown"
    list[12] = daynight

    #For collision type the American dataset uses enumeration
again. We transform
    #the American values to be consistent with the Australian
ones. However,
    #the two datasets had different types of observations
between them for this
    #column, so any observation that was not common between
the two datasets ended
    #up not being included in the final dataset.
    collision = list[13]

    if collision == "1":
        collision = "Rear End"
    elif collision == "2":
        collision = "Head On"
    elif collision  == "4":
        collision = "Right Angle"
    elif collision == "5" or collision == "6":
        collision = "Side Swipe"
    elif collision == "7":
        collision = "Hit Object on Road"
    elif collision == "8":
        collision = "Hit Pedestrian"
    elif collision == "9":
        collision = "Other"
    list[13] = collision
    #For severity the American one again used a numbering 
system, and we also
    #convert some Australian values to be nicer to the eyes. 
Again, since the two
    #datasets did not share the same type of observations we 
disregarded any values
    #that weren't common beween the two datasets.
    severity = list[14]

    if severity == "1: PDO":
        severity = "Not Injured"
    elif severity == "2: MI":
        severity = "Minor Injury"
    elif severity == "3: SI":
        severity = "Major Injury"
    elif severity == "4: Fatal":
        severity = "Fatal"
    elif severity == "0":
        severity = "Not Injured"
    elif severity == "1":
        severity = "Fatal"
    elif severity == "2":
        severity = "Major Injury"
    elif severity == "3":
        severity = "Minor Injury"
    elif severity == "4":
        severity = "Minor Injury"
    list[14] = severity

    #The American data uses bianry for drug presence
    drugs = list[16].strip() #We strip here as it has a 
newline character, being
    #the last value of the row.
    if not drugs:
        drugs = "N"
    if drugs == "0":
        drugs = "N"
    elif drugs == "1":
        drugs = "Y"
    list[16] = drugs

    '''This begins the part of code for checking for incorrect 
values'''

    #Check alcohol column is all the values we want and delete 
values which are
    #not what we expect
    alcohol = list[15]
    if not(alcohol == "Y" or alcohol == "N"):
        flag = False

    #This ensures that all count values are int values and are
not decimals.
    if int(list[1]) != float(list[1]) or int(list[2]) !=
float(list[2]) or int(list[3]) != float(list[3]) or
int(list[4]) != float(list[4]):
        flag = False



    #TotalUnits should be >= 0 and should be of type int
    try:
        totalUnits = int(list[1])
        if not(totalUnits >= 0):
            flag = False
    except:
        flag = False

    #TotalFatalities should be >= 0 and an int
    try:
        totalFatalities = int(list[2])
        if not(totalFatalities >= 0):
            flag = False

    except:
        flag = False




    #Major Injury count is the same deal
    try:
        totalMAjury = int(list[3])
        if not(totalMAjury >= 0):
            flag = False

    except:
        flag = False


    #Minor injury count is the same deal
    try:
        totalMinInjury = int(list[4])
        if not(totalMinInjury >= 0):
            flag = False

    except:
        flag = False
    #CrashYear should be of 2017 as that is what we chose our
dataset to be in
    crashyear = list[5]
    if not(crashyear == "2017"):
        flag = False


    #CrashMonth should be one of the possible months.
    month = list[6]
    if not(month in monthList):
        flag = False


    #Day should be one of the possible days.
    day = list[7]
    if not(day in dayList):
        flag = False


    #The hours for the time should be <= 12, minutes should be
<= 59 and
    #it should be am or pm and hour and minute should be of
type int and all greater than 0
    time = list[8]
    timeList = time.split()
    hourminuteList = timeList[0].split(":")
    try:
        hour = int(hourminuteList[0])
        minute = int(hourminuteList[1])
        endThing = timeList[1].lower()

        if hour > 12 or hour < 0 or minute > 59 or minute < 0:

            flag = False
        if not(endThing == "am" or endThing == "pm"):

            flag = False
    except:

        flag = False

    #Speedlimit should be > 0 and an int. It should also be in
increments of 10.
    try:
        speed = int(list[9])
        if not(speed > 0) or speed % 10 != 0:

            flag = False
    except:

        flag = False
    #Road Moisture we check if it is dry or wet
    moisture = list[10]
    if not(moisture == "Dry") and not(moisture == "Wet"):
        flag = False


    #Check if weather is "Y" or "N" (it also removes NA 
values)
    weather = list[11]
    if not(weather == "Y" or weather == "N"):
        flag = False


    #Check that light is either Daylight or Night (removes 
unknown values)
    light = list[12]
    if not(light == "Night" or light == "Daylight"):
        flag = False


    #Check the crash type is one of the expexcted ones. This 
doesn't remove
    #other and unknown though which we may want to do.
    collision = list[13]
    if not(collision in collisionList):
        flag = False



    #Check that drugs is either "Y" or "N"
    drugs = list[16]
    if not(drugs == "Y" or drugs == "N"):
        flag = False

    list[16] = drugs + "\n"#Add the newline character back to 
drugs now.

    #We make sure that severity is one of the expected ones.
    severity = list[14]
    if not(severity in severityList):
        flag = False



    return (flag, list)

file = open("final.csv", "w+")
csv_readerSA = csv.reader(open("2017_DATA_SA_Crash.csv"))
```

```
first_line = True

#I will first sort and filter through the SA dataset.
for line in csv_readerSA:
    if first_line:
        header = "Nation, Total Units, Total Fatalities, Total
Serious Injuries, Total Minor Injuries, Year, Month, Day,
Time, Speed Limit, Moisture, Adverse Weather Conditions,
DayNight,Crash Type, Crash Severity, Alcohol Related ,Drug
Related\n"
        file.write(header)
        first_line = False
    else:
        nation = "AUS"
        alcohol = line[28]
        totalUnits = line[5]
        totalFatalities = line[7]
        totalCasualties = line[6]
        totalSeriousInjury = line[8]
        totalMinorInjury = line[9]
        year = line[10]
        month = line[11]
        day = line[12]
        time = line[13]
        speedLimit = line[14]
        positionType = line[15]

        roadMoisture = line[20]
        weather = line[21]
        #lightCondition = line Actually I can't seem to find
this one in the dataset, maybe we can replace it with the
daytime or nighttime one?
        crashType = line[23]
        crashSeverity = line[26]
        trafficControl = line[27]
        drugs = line[29]
        lightCondition = line[22]



        if not drugs:
            drugs = "N"

        #The dataset keeps no alcohol information as blank
spaces, but it is more useful to us as "N".
        if not alcohol:
            alcohol = "N"

        row =
"{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n".format(
nation, totalUnits, totalFatalities, totalSeriousInjury,
totalMinorInjury, year, month, day, time, speedLimit,
roadMoisture, weather, lightCondition, crashType,
crashSeverity, alcohol, drugs)
        list = row.split(",")#first I turn the row string into
a list for use with
        #the cleaning function.

        tuple = cleaningFunction(list)#We assign a value to
the tuple that the
        #cleaning function returns.
        flag = tuple[0]#In the cleaning function, there was a
boolean value which
        #determines if there were any incorrect values. It is
set to true, by
        #default, and if the function encounters any incorrect
values it is set to false.

        list = tuple[1]#We retrieve the newly transformed list
that now accounts
        #for inconsistent values due to the cleaningFunction

        #If there were no incorrect values, then we right this
particular row to
        #the file by converting the list back to a string.
        if flag:
            file.write(", ".join(list))

#We apply similar logic to the American dataset.
csv_readerA = csv.reader(open("bf8b3c7e-8d60-40df-9134-
21606a451c1a.csv"))
first_line = True

for line in csv_readerA:
    if first_line:
        pass

        first_line = False
    else:
        Nation = "US"
        alcohol = line[127]
        totalUnits = line[21]
        totalFatalities = line[32]
        totalSeriousInjury = line[34]
        totalMinorInjury = line[36]
        year = line[5]

        month = line[6]
        day = line[7]
        time = line[8]

        #I account for the fact that the American dataset is
in mph and round it up
```

```
        #to the nearest 10.
        speedLimit = line[184]
        if speedLimit != "":
            speedLimit = round(int(speedLimit) * 1.60934)
        speedLimit = str(speedLimit)
        if len(speedLimit) >= 2:
            if int(speedLimit[-1]) < 5:
                speedLimit = speedLimit[0:len(speedLimit) - 1]
+ "0"
            elif int(speedLimit[-1]) >= 5:
                speedLimit =
str(int(speedLimit[0:len(speedLimit) - 1]) + 1) + "0"



        roadMoisture = line[91]
        weather = line[11]

        crashType = line[13]
        crashSeverity = line[52]

        drugs = line[165]
        lightCondition = line[10]


        row =
"{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{}\n".format(
Nation, totalUnits, totalFatalities, totalSeriousInjury,
totalMinorInjury, year, month, day, time, speedLimit,
roadMoisture, weather, lightCondition, crashType,
crashSeverity, alcohol, drugs)
        list = row.split(",")

        tuple = cleaningFunction(list)
        flag = tuple[0]
        list = tuple[1]


        if flag:
            file.write(", ".join(list))

file.close()
```

Analysis

Daily

```
import csv
is_first_line = True
countMonday = 0
countTuesday = 0
countWednesday = 0
countThursday = 0
countFriday = 0
countSaturday = 0
countSunday = 0
countWeekday = 0
countWeekend = 0
countWeekdayAUS = 0
countWeekdayUS = 0
countWeekendAUS = 0
countWeekendUS = 0


with open('final.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for row in readCSV:
       if is_first_line:
            is_first_line = False
       else:
            if row[7] is 'Monday':
                countMonday += 1
                countWeekday += 1
                if row[0] is 'AUS':
                    countWeekdayAUS += 1
                else:
                    countWeekdayUS +=1
            elif row[7] = 'Tuesday':
                countTuesday += countTuesday + 1
                countWeekday += 1
                if row[0] is 'AUS':
                    countWeekdayAUS += 1
                else:
                    countWeekdayUS +=1
            elif row[7] = 'Wednesday':
                countWednesday += countWednesday + 1
                countWeekday += 1
                if row[0] is 'AUS':
                    countWeekdayAUS += 1
                else:
                    countWeekdayUS +=1
            elif row[7] = 'Thursday':
                countThursday += countThursday + 1
                countWeekday += 1
                if row[0] is 'AUS':
                    countWeekdayAUS += 1
                else:
                    countWeekdayUS +=1
            elif row[7] = 'Friday':
                countFriday += countFriday + 1
                countWeekday += 1
                if row[0] is 'AUS':
                    countWeekdayAUS += 1
                else:
                    countWeekdayUS +=1
            elif row[7] = 'Saturday':
                countSaturday += countSaturday + 1
                countWeekend += 1
                if row[0] is 'AUS':
                    countWeekendAUS += 1
                else:
                    countWeekendUS +=1
            elif row[7] = 'Sunday':
                countSunday += countSunday + 1
                countWeekend += 1
                if row[0] is 'AUS':
                    countWeekendAUS += 1
                else:
                    countWeekendUS +=1

print("number of crashes on weekdays are:", countWeekday)
print("number of crashes on weekends are:", countWeekend)
print("number of crashes on Mondays are:", countMonday)
print("number of crashes on Tuesdays are:", countTuesday)
print("number of crashes on Wednesdays are:", countWednesday)
print("number of crashes on Thursdays are:", countThursday)
print("number of crashes on Fridays are:", countFriday)
print("number of crashes on Saturdays are:", countSaturday)
print("number of crashes on Sundays are:", countSunday)
print("number of crashes in Australia on weekdays are:",
countWeekdayAUS)
print("number of crashes in United States on weekdays are:",
countWeekdayUS)
print("number of crashes in Australia on weekends are:",
countWeekendAUS)
print("number of crashes in United States on weekends are:",
countWeekendUS)
```

Analysis(aggregate)

```
#This program finds the mean amount of units for crashes in
each nation corresponding to positive and negative weather
conditions.
import statistics;
import pandas as pd
import numpy as np
```

```
data_by_nation = {}
is_first_line = True
file = open("analysis.txt","w+")
print("Analysis 1\n")
file.write("Analysis 1\n\n")
for row in open("final.csv"):
  if is_first_line:
    is_first_line = False

  else:
    #We extract the values for all the necessary information
from the file.
    values = row.split(",")
    nation = values[0]
    weather = values[11]
    totalUnits = int(values[1])
    #We create a key which contains information on the nation
and the weather conditions.
    dict_key = (nation, weather)

    if dict_key in data_by_nation:
        data_by_nation[dict_key].append(totalUnits)
    else:
        data_by_nation[dict_key] = [totalUnits]

#We go through all the nation and weather combinations and
find the mean.
print("Mean for each nation and corresponding weather
condition:\n")
file.write("Mean for each nation and corresponding weather
condition:\n\n")
for key in data_by_nation:
    print("{}: {}\n".format(key,
statistics.mean(data_by_nation[key])))
    file.write("{}: {}\n".format(key,
statistics.mean(data_by_nation[key])))

#These combined lists concatenate the two lists for each
nation and then later finds the mean of them.
combinedUS = data_by_nation[('US', ' N')] +
data_by_nation[('US', ' Y')]
combinedAUS = data_by_nation[('AUS', ' N')] +
data_by_nation[('AUS', ' Y')]
print()
file.write("\n\n")
print("Overall mean for US:
{}\n".format(statistics.mean(combinedUS)))
file.write("Overall mean for US:
{}\n".format(statistics.mean(combinedUS)))
```

```
print("Overall mean for AUS:
{}\n".format(statistics.mean(combinedAUS)))
file.write("Overall mean for AUS:
{}\n".format(statistics.mean(combinedAUS)))
file.write("\n\n")

#------------------------------||--||-------------------------
-----------------
#------------------------------||--||-------------------------
-----------------

print("Analysis 2\n")
file.write("Analysis 2\n\n")

#Read our final csv file
df = pd.read_csv("final.csv")

#I grouped the values based on two conditions:
#Whether it is US or AUS and whether that day was rainy or not
#The I separated the table to minor and major injuries
#in order to calculate the number of minor and major injures
separately

major = df.sort_values(["Nation"]).groupby(["Nation", "Adverse
Weather Conditions"], sort=False)["Total Serious Injuries"]
minor = df.sort_values(["Nation"]).groupby(["Nation", "Adverse
Weather Conditions"], sort=False)["Total Minor Injuries"]

#I use the sum function to sum up the injuries
num_major = major.sum()
num_minor = minor.sum()

#Here I created a new column using the np.where function
#where if there was no minor and major injury
#it returned 1 and 0 if there was
df['No Injury'] = np.where((df['Total Serious
Injuries']==0)&(df['Total Minor Injuries']==0), 1,0)

#Here I used the sum function to calculate how many crashes
there were
#where injury did not result
no_inj = df.sort_values(["Nation"]).groupby(["Nation",
"Adverse Weather Conditions"], sort=False)["No Injury"].sum()

#create dictionaries
num_minor_dict = num_minor.to_dict()
num_major_dict = num_major.to_dict()
num_noinj_dict =no_inj.to_dict()
file.write("Number of major injuries based on Nation and
weather condition:\n\n")
```

```
print("Number of major injuries based on Nation and weather
condition:\n")
for i in num_major_dict:
    print(str(i)+" : "+str(num_major_dict[i]))
    file.write(str(i)+" : "+str(num_major_dict[i])+"\n")
file.write("\n\n")
print("\n")
print("Number of minor injuries based on Nation and weather
condition:\n")
file.write("Number of minor injuries based on Nation and
weather condition:\n\n")
for i in num_minor_dict:
    print(str(i)+" : "+str(num_minor_dict[i]))
    file.write(str(i)+" : "+str(num_minor_dict[i])+"\n")
print("\n")
file.write("\n\n")
print("Number of no injuries based on Nation and weather
condition:\n")
file.write("Number of no injuries based on Nation and weather
condition:\n\n")
for i in num_noinj_dict:
    print(str(i)+" : "+str(num_noinj_dict[i]))
    file.write(str(i)+" : "+str(num_noinj_dict[i])+"\n")
print("\n")
file.write("\n\n")
print("Percentage of major injuries based on Nation and
weather condition:\n")
file.write("Percentage of major injuries based on Nation and
weather condition:\n\n")
for i in num_major_dict:
    file.write(str(i)+" :
"+str(round(num_major_dict[i]*100/(num_minor_dict[i]+num_noinj
_dict[i]+num_major_dict[i]),2))+"\n")
    print(str(i)+" :
"+str(round(num_major_dict[i]*100/(num_minor_dict[i]+num_noinj
_dict[i]+num_major_dict[i]),2)))
print("\n")
file.write("\n\n")
print("Percentage of minor injuries based on Nation and
weather condition:\n")
file.write("Percentage of minor injuries based on Nation and
weather condition:\n\n")
for i in num_minor_dict:
    #print(num_minor_dict[i])
    file.write(str(i)+" :
"+str(round(num_minor_dict[i]*100/(num_noinj_dict[i]+num_minor
_dict[i]+num_major_dict[i]),2))+"\n")
    print(str(i)+" :
"+str(round(num_minor_dict[i]*100/(num_noinj_dict[i]+num_minor
_dict[i]+num_major_dict[i]),2)))
print("\n")
```

```
file.write("\n\n")
print("Percentage of injuries based on Nation and weather
condition:\n")
file.write("Percentage of injuries based on Nation and weather
condition:\n\n")
for i in num_minor_dict:
    a =
(num_major_dict[i]+num_minor_dict[i])*100/(num_minor_dict[i]+n
um_major_dict[i]+num_noinj_dict[i])
    print(str(i)+" : "+str(round(a,2)))
    file.write(str(i)+" : "+str(round(a,2))+"\n")

file.close()
```

Analysis(Daily)

```
import csv
is_first_line = True
countMonday = 0
countTuesday = 0
countWednesday = 0
countThursday = 0
countFriday = 0
countSaturday = 0
countSunday = 0
countWeekday = 0
countWeekend = 0
countWeekdayAUS = 0
countWeekdayUS = 0
countWeekendAUS = 0
countWeekendUS = 0


for row in open("final.csv"):
       if is_first_line:
            is_first_line = False
       else:
            values = row.split(",")
            if values[7] == " Monday":
               countMonday += 1
               countWeekday += 1
               if values[0] == "AUS":
                   countWeekdayAUS += 1
               else:
                    countWeekdayUS +=1
            if values[7] == " Tuesday":
               countTuesday += 1
               countWeekday += 1
               if values[0] == "AUS":
                   countWeekdayAUS += 1
               else:
                    countWeekdayUS +=1
            if values[7] == " Wednesday":
               countWednesday += 1
               countWeekday += 1
               if values[0] == "AUS":
                   countWeekdayAUS += 1
               else:
                   countWeekdayUS +=1
            if values[7] == " Thursday":
               countThursday += 1
               countWeekday += 1
               if values[0] == "AUS":
                   countWeekdayAUS += 1
               else:
                   countWeekdayUS +=1
            if values[7] == " Friday":
               countFriday += 1
               countWeekday += 1
               if values[0] == "AUS":
                   countWeekdayAUS += 1
               else:
                   countWeekdayUS +=1
            if values[7] == " Saturday":
               countSaturday += 1
               countWeekend += 1
               if values[0] == "AUS":
                   countWeekendAUS += 1
               else:
                    countWeekendUS +=1
            if values[7] == " Sunday":
               countSunday += 1
               countWeekend += 1
               if values[0] == "AUS":
                   countWeekendAUS += 1
               else:
                   countWeekendUS +=1

print("number of crashes on weekdays are:", countWeekday)
print("number of crashes on weekends are:", countWeekend)
print("number of crashes on Mondays are:", countMonday)
print("number of crashes on Tuesdays are:", countTuesday)
print("number of crashes on Wednesdays are:", countWednesday)
print("number of crashes on Thursdays are:", countThursday)
print("number of crashes on Fridays are:", countFriday)
print("number of crashes on Saturdays are:", countSaturday)
print("number of crashes on Sundays are:", countSunday)
print("number of crashes in Australia on weekdays are:",
countWeekdayAUS)
print("number of crashes in United States on weekdays are:",
countWeekdayUS)
```

```
print("number of crashes in Australia on weekends are:",
countWeekendAUS)
print("number of crashes in United States on weekends are:",
countWeekendUS)
```