Parameters for estimation

Example: Location and variability Estimates of Population and Murder Rates.

data set containing population and murder rates (in units of murders per 100,000 people per year) for each US state (2010 Census)

Compute the mean, trimmed mean, median, standard deviation, interquartile range, and mean absolute deviation from the mean for the population using Python/R.

In [100]:

```
#Read the dataset using pandas method of data frame.
import pandas as pd
state = pd.read_csv('state.csv')
print(state)
```

```
           State  Population  Murder.Rate Abbreviation
0        Alabama     4779736          5.7           AL
1         Alaska      710231          5.6           AK
2        Arizona     6392017          4.7           AZ
3       Arkansas     2915918          5.6           AR
4     California    37253956          4.4           CA
5       Colorado     5029196          2.8           CO
6    Connecticut     3574097          2.4           CT
7       Delaware      897934          5.8           DE
8        Florida    18801310          5.8           FL
9        Georgia     9687653          5.7           GA
10        Hawaii     1360301          1.8           HI
11         Idaho     1567582          2.0           ID
12      Illinois    12830632          5.3           IL
13       Indiana     6483802          5.0           IN
14          Iowa     3046355          1.9           IA
15        Kansas     2853118          3.1           KS
16      Kentucky     4339367          3.6           KY
17     Louisiana     4533372         10.3           LA
18         Maine     1328361          1.6           ME
19      Maryland     5773552          6.1           MD
20 Massachusetts     6547629          2.0           MA
21      Michigan     9883640          5.4           MI
22     Minnesota     5303925          1.6           MN
23   Mississippi     2967297          8.6           MS
24      Missouri     5988927          6.6           MO
25       Montana      989415          3.6           MT
26      Nebraska     1826341          2.9           NE
27        Nevada     2700551          6.0           NV
28 New Hampshire     1316470          0.9           NH
29    New Jersey     8791894          3.9           NJ
30    New Mexico     2059179          4.8           NM
31      New York    19378102          3.1           NY
32 North Carolina     9535483          5.1           NC
33  North Dakota      672591          3.0           ND
34          Ohio    11536504          4.0           OH
35      Oklahoma     3751351          4.5           OK
36        Oregon     3831074          2.0           OR
37  Pennsylvania    12702379          4.8           PA
38  Rhode Island     1052567          2.4           RI
39 South Carolina     4625364          6.4           SC
40  South Dakota      814180          2.3           SD
41     Tennessee     6346105          5.7           TN
42         Texas    25145561          4.4           TX
43          Utah     2763885          2.3           UT
44       Vermont      625741          1.6           VT
45      Virginia     8001024          4.1           VA
46    Washington     6724540          2.5           WA
47 West Virginia     1852994          4.0           WV
48     Wisconsin     5686986          2.9           WI
```

```
49          Wyoming       563626          2.7           WY
```

To compute mean and median in Python we can use the pandas methods of the data frame. Mean: The Mean is sum of all values divided by the number of values. calculate mean of population and Murder rate columns from the dataset.

In [101]:

```
#calculate mean of population
population_mean = state['Population'].mean()
print(f' Mean of state population is :  {population_mean}')
```

```
 Mean of state population is :  6162876.3
```

In [102]:

```
#calculate mean of Murder rate
Murder_mean = state['Murder.Rate'].mean()
print(f' Mean of Murder rate is :  {Murder_mean}')
```

```
 Mean of Murder rate is :  4.066
```

Median:The median is the middle number on a sorted list of the data. If there is an even number of data values, the middle value is one that is not actually in the data set, but rather the average of the two values that divide the sorted data into upper and lower halves. For better understanding take the first 9 samples of the original dataset for calculating the median when dataset has odd number of elemnets. Next, first 10 samples when dataset has even number of elements.

sorted data = { 2.4, 2.8, 4.4, 4.7, 5.6,5.6,5.7,5.8,5.8}

In [103]:

```
#sample dataset from original dataset
state_9 = state[:9]
state_9
```

Out[103]:

	State	Population	Murder.Rate	Abbreviation
0	Alabama	4779736	5.7	AL
1	Alaska	710231	5.6	AK
2	Arizona	6392017	4.7	AZ
3	Arkansas	2915918	5.6	AR
4	California	37253956	4.4	CA
5	Colorado	5029196	2.8	CO
6	Connecticut	3574097	2.4	CT
7	Delaware	897934	5.8	DE
8	Florida	18801310	5.8	FL

In [104]:

```
#median using odd numbers of elements.
population_median = state_9['Population'].median()
print(f' Mean of state population is :  {population_median}')
```

```
 Mean of state population is :  4779736.0
```

In [105]:

```
#sample dataset from original dataset
state_10 = state[:10]
state_10
```

Out[105]:

Out[105]:

	State	Population	Murder.Rate	Abbreviation
0	Alabama	4779736	5.7	AL
1	Alaska	710231	5.6	AK
2	Arizona	6392017	4.7	AZ
3	Arkansas	2915918	5.6	AR
4	California	37253956	4.4	CA
5	Colorado	5029196	2.8	CO
6	Connecticut	3574097	2.4	CT
7	Delaware	897934	5.8	DE
8	Florida	18801310	5.8	FL
9	Georgia	9687653	5.7	GA

In [106]:

```
#median using even number of data values
population_median = state_10['Population'].median()
print(f' Mean of state population is :  {population_median}')
```

```
 Mean of state population is :  4904466.0
```

Weighted mean: calculated by multiplying each data value xi by a user specified weight wi and dividing their sum by the sum of the weights. When making decisions when certain factors are more crucial than others, weighted mean might be useful. Create two lists, one is for values and another is for weights. Create a user defined function name weighted_mean() with two parameters, value and weight.Use for loop in a function to retrive all values from list and weights.

In [107]:

```
def weighted_mean(number, weights):
  return sum(x * y for x, y in zip(number, weights)) / sum(weights)


list1 = [15, 60, 40]
list2 = [6, 5, 2]

print("Original list of elements:")
print(list1)
print(list2)

print("\nWeighted average of the said two list of numbers:")
print(weighted_mean(list1, list2))
```

```
Original list of elements:
[15, 60, 40]
[6, 5, 2]

Weighted average of the said two list of numbers:
36.15384615384615
```

Trimmed mean: : Calculated by dropping a fixed number of sorted values at each end and then taking an average of remaining values. Compute 20% trimmed mean for an array of data:

In [108]:

```
from scipy import stats

#define data
data = [6.8, 8.1, 8.3, 9.1,9.3]

#calculate 10% trimmed mean
trimean=stats.trim_mean(data, 0.2)
```

```
print(f' trimmed mean of the above data is equal to : {trimean}')
```

```
 trimmed mean of the above data is equal to : 8.5
```

Percentile: In statistics refers to a score's position in relation to other scores from the same set.In statistics, percentiles are used to provide a numerical representation of the value that a specific percentage of values are lower than.

In [109]:

```
from scipy import stats
std_score = [67, 70, 75, 76, 77, 78, 80, 83, 85, 87, 88, 89, 90, 93, 95]
stats.percentileofscore(std_score, 88)
```

Out[109]:

```
73.33333333333333
```

In [110]:

```
stats.percentileofscore([1, 2, 3, 3, 4], 3, kind='mean')
```

Out[110]:

```
60.0
```

Example: Let's say we have an array of the ages of all the people that lives in a street. What is the 75. percentile?

In [111]:

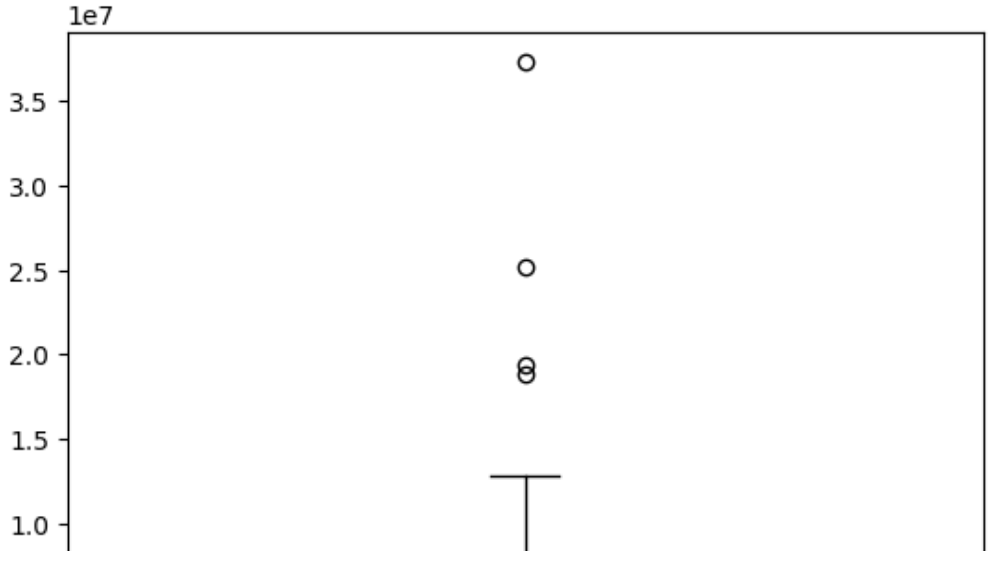
```
import numpy

ages1 = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]


x = numpy.percentile(ages1, 75)

print(x)
```

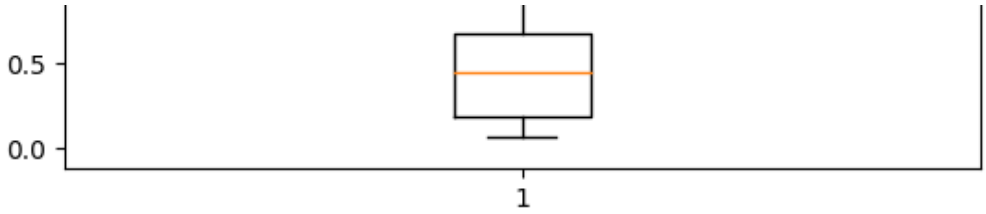
```
43.0
```

Outlier: outlier is an untypical observed data point in a given distribution of data points. We can visualise outliers in our dataset using Histogram, boxplot, and scatter plot

In [112]:

```
#outlier visualisation using boxplot
import matplotlib.pyplot as plt
plt.boxplot(state['Population'])
fig = plt.figure(figsize =(10, 7))
plt.show()
```

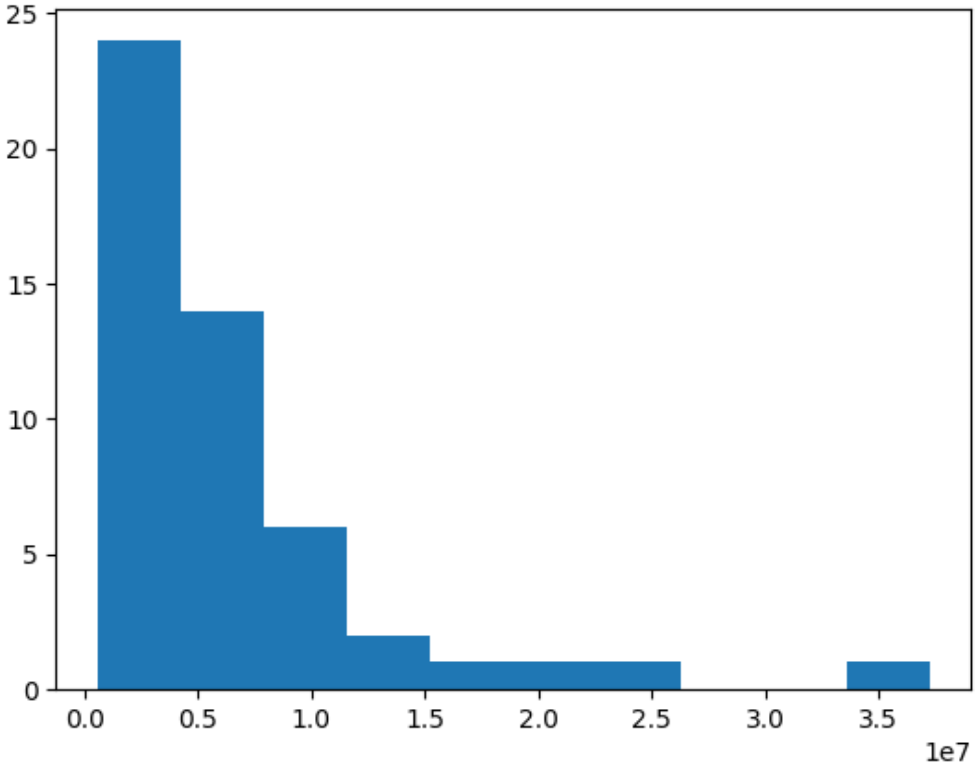





```
<Figure size 1000x700 with 0 Axes>
```

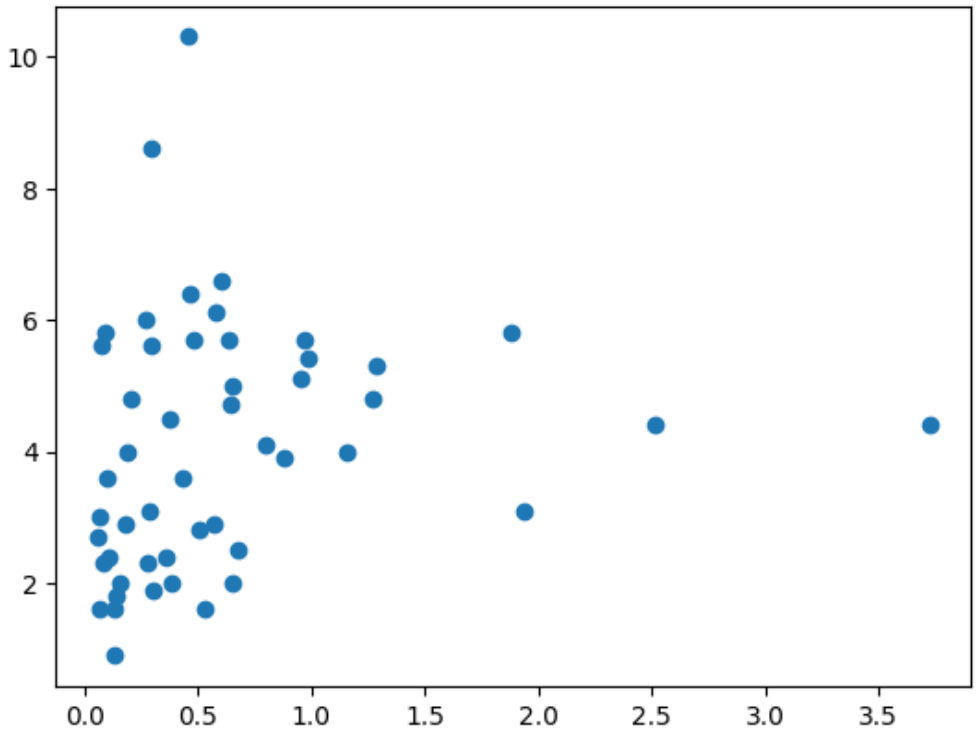
In [113]:

```
#outlier visualisation using histogram
plt.hist(state['Population'])
plt.show()
```



In [114]:

```
plt.scatter(state['Population'],state['Murder.Rate'])
plt.show()
```



1e7

In [ ]:

```
Variance: The sum of squared deviations from the mean divided by n - 1 where n is the nu
mber of data values. Find the mean
and each score's deviation from the mean. Next, find the sum of squared for each deviati
on from the mean and divide it
by number of samples.
```

In [84]:

```
import statistics

var=statistics.variance([46,69,32,60,52,41])

print(var)
```

```
177.2
```

Standard deviation: is the square root of the variance.

In [119]:

```
import math
std_dev = math.sqrt(var)
std_dev
```

Out[119]:

```
13.311649033834989
```

Mean absolute deviation: The mean of the absolute values of the deviation from the mean. Calculate mean of the data, calculate absolute deviation from the mean for each sample,calculate the average of mean absolute value of deviation.

In [118]:

```
from statistics import mean

numbers = [1,4,4]
mean_value = mean(numbers)
mean_absolute_deviation = mean([abs(number-mean_value) for number in numbers])
print(mean_absolute_deviation)
```

```
1.3333333333333333
```

Median absolute deviation: The median of the absolute values of the deviation from the median. Calculate median of the data, calculate absolute deviation from the median for each sample.

In [117]:

```
from statistics import median

numbers = [1,4,4]
median_value = median(numbers)
median_absolute_deviation = median([abs(number-median_value) for number in numbers])
print(median_absolute_deviation)
```

```
0
```