# Homework 1

September 30, 2019

## 1 Homework 1 - Math 565

### 1.0.1 Richard Vargas

In the cell below, I import all libraries needed. Pandas contains the data structure Data Frame which makes structuring data easier Numpy allows for operations on the data Matplotlib is used for data visualization.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

## 2 Problem 6.1-4

Here I import the homework data, give a column name, and display some of the data.

[2]:
```
Problem6_1_4data = pd.read_csv('E6_1-04.txt',sep=" ", header=None)
Problem6_1_4data.rename(columns={0:"Melting_Pt"},inplace=True)
Problem6_1_4data.head()
```

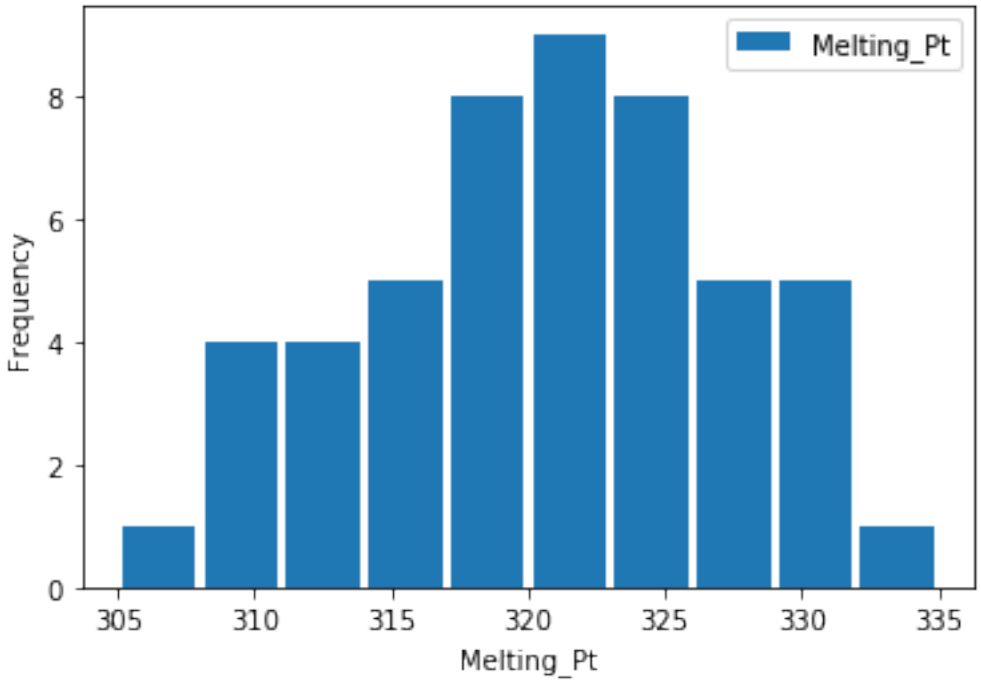
[2]:

| | Melting_Pt |
|---|---|
| 0 | 320 |
| 1 | 326 |
| 2 | 325 |
| 3 | 318 |
| 4 | 322 |

### 2.1 Part a

A histogram, which auto generates bin size.

[3]:
```
Problem6_1_4data.plot(kind='hist',rwidth=0.9)
plt.xlabel(Problem6_1_4data.columns.values[0])
plt.show()
```




### 2.2 Part b

Calculation of mean and standard deviation.

[4]:
```
dataMean = np.mean(Problem6_1_4data.values)
print("The mean of the data is",dataMean)

dataStd = np.std(Problem6_1_4data.values)
print("The standard deviation is",dataStd)
```

```
The mean of the data is 320.1
The standard deviation is 6.682065548915245
```

### 2.3 Part c

Below is the amount of elements 1 and 2 standard deviations away

[5]:
```
sd1Count = 0
sd2Count = 0
for c in Problem6_1_4data.values:
    if(c <= dataMean+2*dataStd and c >= dataMean-2*dataStd):
        sd2Count+=1
        if(c <= dataMean+dataStd and c >= dataMean-dataStd):
            sd1Count+=1

print("The amount of elements within 1 standard deviation is",sd1Count)
```

```
print("The amount of elements within 2 standard deviation is",sd2Count)
```

```
The amount of elements within 1 standard deviation is 31
The amount of elements within 2 standard deviation is 48
```

# 3 Problem 6.1-5

[6]:
```
Problem6_1_5data = pd.read_csv('E6_1-05.txt',sep=" ", header=None)
Problem6_1_5data.rename(columns={0:"Successive_Bets"},inplace=True)
Problem6_1_5data.head()
```

[6]:

| | Successive_Bets |
|---|---|
| 0 | 23 |
| 1 | 127 |
| 2 | 877 |
| 3 | 65 |
| 4 | 101 |

## 3.1 Part a

Finding sample mean and sample standard deviation

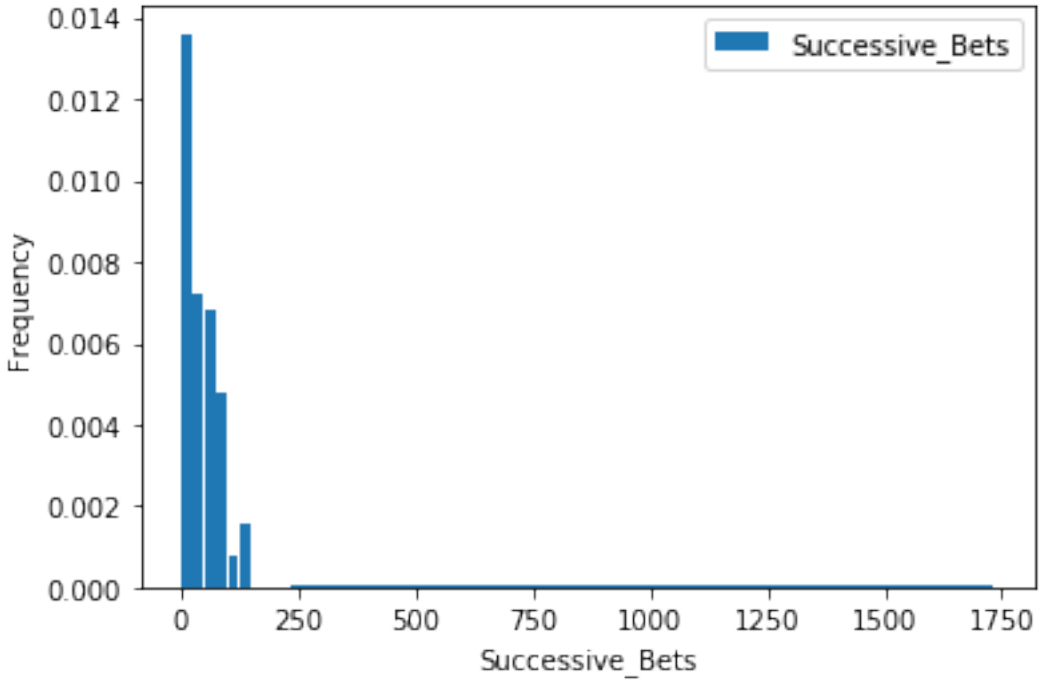
[7]:
```
dataMean = np.mean(Problem6_1_5data.values)
print("The mean of the data is",dataMean)

dataStd = np.std(Problem6_1_5data.values)
print("The standard deviation is",dataStd)
```

```
The mean of the data is 112.12
The standard deviation is 230.19788356976701
```

## 3.2 Part b

Histogram of the data.

[8]:
```
Problem6_1_5data.
 ↪plot(kind='hist',density=True,bins=[0,25,50,75,100,125,150,Problem6_1_5data.
 ↪values.max()],rwidth=0.9)
plt.xlabel(Problem6_1_5data.columns.values[0])
plt.show()
```




### 3.3 Part c

Locating x bar, and the 1, 2 and 3, standard deviations.

[9]:
```
print("x bar is",dataMean)
print("x bar +/- one standard deviation␣
 ↪is",dataMean+dataStd,"and",dataMean-dataStd)
print("x bar +/- two standard deviation␣
 ↪is",dataMean+2*dataStd,"and",dataMean-2*dataStd)
print("x bar +/- three standard deviation␣
 ↪is",dataMean+3*dataStd,"and",dataMean-3*dataStd)
```

```
x bar is 112.12
x bar +/- one standard deviation is 342.317883569767 and -118.07788356976701
x bar +/- two standard deviation is 572.5157671395341 and -348.275767139534
x bar +/- three standard deviation is 802.7136507093011 and -578.4736507093011
```

### 3.4 Part d

This data is heavily skewed, so using the median would be better than the mean, since the median is more robust to skewed data.

# 4 Problem 6.1-8

[10]:
```
Problem6_1_8data = pd.read_csv('E6_1-08.txt',sep=" ", header=None)
Problem6_1_8data.rename(columns={0:"Mirror_Weight(g)"},inplace=True)
Problem6_1_8data.head()
```

[10]:

| | Mirror_Weight(g) |
|---|---|
| 0 | 3.968 |
| 1 | 3.534 |
| 2 | 4.032 |
| 3 | 3.912 |
| 4 | 3.572 |

## 4.1 Part a

Create a frequency distribution

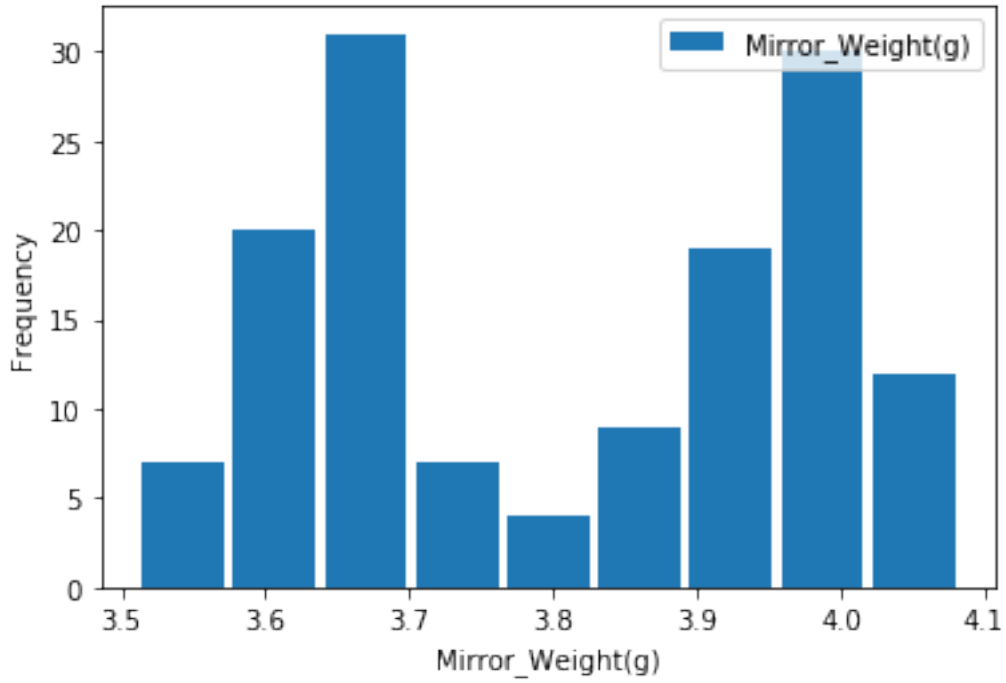
[11]:
```
dataRange=Problem6_1_8data.values.max()-Problem6_1_8data.values.min()
classSize=(dataRange/8).round(decimals=3)
currentNum = Problem6_1_8data.values.min()
dataBins = {}
while(currentNum<Problem6_1_8data.values.max()):
    dataBins[currentNum.round(decimals=3)]=0
    currentNum += classSize
for a in Problem6_1_8data.values:
    for x in dataBins:
        if (a>=x and a<x+classSize):
            dataBins[x]+=1
            break
        if (x==4.079):
            dataBins[x]+=1

freqDist=pd.DataFrame(list(dataBins.items()))
freqDist.rename(columns={0:"class",1:"frequency"},inplace=True)
print(freqDist)
```

| | class | frequency |
|---|---|---|
| 0 | 3.511 | 8 |
| 1 | 3.582 | 27 |
| 2 | 3.653 | 29 |
| 3 | 3.724 | 3 |
| 4 | 3.795 | 8 |
| 5 | 3.866 | 17 |
| 6 | 3.937 | 28 |
| 7 | 4.008 | 17 |
| 8 | 4.079 | 2 |

### 4.2 Part b

Draw histogram of the data

[12]:
```
Problem6_1_8data.plot(kind='hist',rwidth=0.9,bins=9)
plt.xlabel(Problem6_1_8data.columns.values[0])
plt.show()
```




### 4.3 Part c

The shape of this graph has two peaks, making it bimodal.

## 5 Problem 6.2-2

[13]:
```
Problem6_2_2data = pd.read_csv('E6_2-02.txt',sep="\t", header=None)
Problem6_2_2data.rename(columns={0:"Baby_Carrot_Wt",1:
 ↪"Reg_Carrot_Wt"},inplace=True)
Problem6_2_2data.head()
```

[13]:

| | Baby_Carrot_Wt | Reg_Carrot_Wt |
|---|---|---|
| 0 | 1.03 | 1.29 |
| 1 | 1.03 | 1.10 |
| 2 | 1.06 | 1.28 |
| 3 | 1.02 | 1.29 |
| 4 | 1.03 | 1.23 |

### 5.1 Part a

Calculate the five-number summary of each set of weights

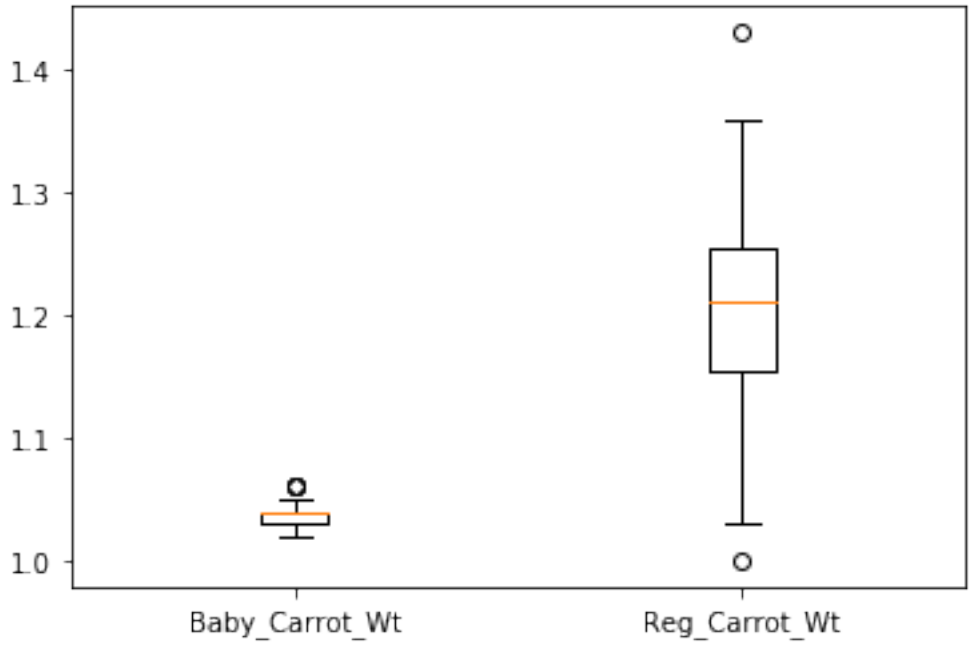
[14]:
```
Baby_Carrot_Quartile = np.percentile(Problem6_2_2data.loc[:,"Baby_Carrot_Wt"].
 ↪values,[25,50,75])
Baby_Carrot_Min, Baby_Carrot_Max = Problem6_2_2data.loc[:,"Baby_Carrot_Wt"].
 ↪values.min(),Problem6_2_2data.loc[:,"Baby_Carrot_Wt"].values.max()
print("The minimum of baby carrots is",Baby_Carrot_Min)
print("The lower quartile is",Baby_Carrot_Quartile[0])
print("The median is",Baby_Carrot_Quartile[1])
print("The upper quartile is",Baby_Carrot_Quartile[2])
print("The maximum is",Baby_Carrot_Max,"\n")

Reg_Carrot_Quartile = np.percentile(Problem6_2_2data.loc[:,"Reg_Carrot_Wt"].
 ↪values,[25,50,75])
Reg_Carrot_Min, Reg_Carrot_Max = Problem6_2_2data.loc[:,"Reg_Carrot_Wt"].values.
 ↪min(),Problem6_2_2data.loc[:,"Reg_Carrot_Wt"].values.max()
print("The minimum of regular carrots is",Reg_Carrot_Min)
print("The lower quartile is",Reg_Carrot_Quartile[0].round(2))
print("The median is",Reg_Carrot_Quartile[1])
print("The upper quartile is",Reg_Carrot_Quartile[2].round(2))
print("The maximum is",Reg_Carrot_Max)
```

```
The minimum of baby carrots is 1.02
The lower quartile is 1.03
The median is 1.04
The upper quartile is 1.04
The maximum is 1.06

The minimum of regular carrots is 1.0
The lower quartile is 1.15
The median is 1.21
The upper quartile is 1.25
The maximum is 1.43
```

### 5.2 Part b

Construct a box plot for each set of weights

[15]:
```
plt.boxplot(Problem6_2_2data.values,labels=Problem6_2_2data.columns)
plt.show()
```




### 5.3 Part c

If the carrots are the same price per package, the regular carrots would be a better purchase, since they are most likely to be heavier and thus contain more.

## 6 Problem 6.2-4

[16]:
```
Problem6_2_4data = pd.read_csv('E6_2-04.txt',sep=" ", header=None)
Problem6_2_4data.rename(columns={0:"Losses($10,000)"},inplace=True)
Problem6_2_4data.head()
```

[16]:

| | Losses($10,000) |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 2 |
| 3 | 3 |
| 4 | 3 |

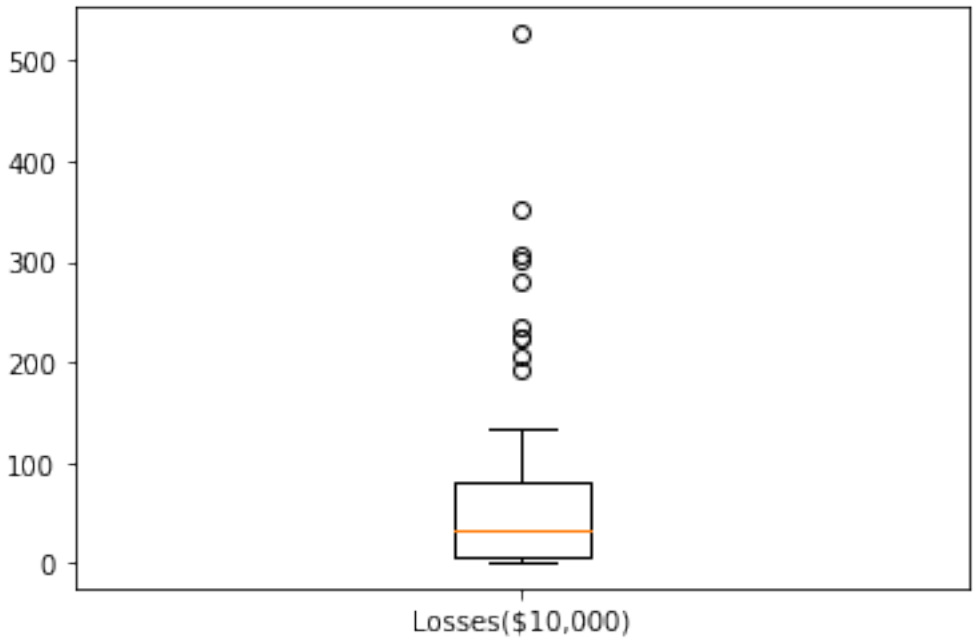
### 6.1 Part a

Create five-number summary of the data and draw a box-and-whisker diagram

[17]:
```
Losses_Quartile = np.percentile(Problem6_2_4data.values,[25,50,75])
Losses_Min, Losses_Max = Problem6_2_4data.values.min(),Problem6_2_4data.values.
 ↪max()
print("The minimum of losses(in thousands) is",Losses_Min)
```

```
print("The lower quartile is",Losses_Quartile[0])
print("The median is",Losses_Quartile[1])
print("The upper quartile is",Losses_Quartile[2])
print("The maximum is",Losses_Max,"\n")

plt.boxplot(Problem6_2_4data.values,labels=Problem6_2_4data.columns)
plt.show()
```

```
The minimum of losses(in thousands) is 1
The lower quartile is 7.0
The median is 32.0
The upper quartile is 80.0
The maximum is 527
```




### 6.2 Part b

Calculate the IQR and the locations of the inner and outer fences

[18]:
```
dataIQR = Losses_Quartile[2]-Losses_Quartile[0]
inner_Fence = dataIQR * 1.5
outer_Fence = dataIQR * 3

print("The IQR of the losses is",Losses_Quartile[1]+dataIQR,␣
 ↪"and",Losses_Quartile[1]-dataIQR)
```

```
print("The inner fences of the losses␣
 ↪is",Losses_Quartile[0]-inner_Fence,"and",Losses_Quartile[2]+inner_Fence)
print("The outer fences of the losses␣
 ↪is",Losses_Quartile[0]-outer_Fence,"and",Losses_Quartile[2]+outer_Fence)
```

```
The IQR of the losses is 105.0 and -41.0
The inner fences of the losses is -102.5 and 189.5
The outer fences of the losses is -212.0 and 299.0
```

### 6.3 Part c

Draw a box plot that shows fences, suspected outliers, and outliers (See Part a)

### 6.4 Part d

Describe the distribution of losses Much of the losses are less than 180,000, with outliers in the 180,000 - 350,000 range. There is one extreme outlier in the $550,000 area

## 7 Problem 6.2-10

[19]:
```
Problem6_2_10data = pd.read_csv('E6_2-10.txt',sep="\t", header=None)
Problem6_2_10data.rename(columns={0:"1976_Pb_Con(ug/m^3)",1:"1977_Pb_Con(ug/
 ↪m^3)"},inplace=True)
Problem6_2_10data.head()
```

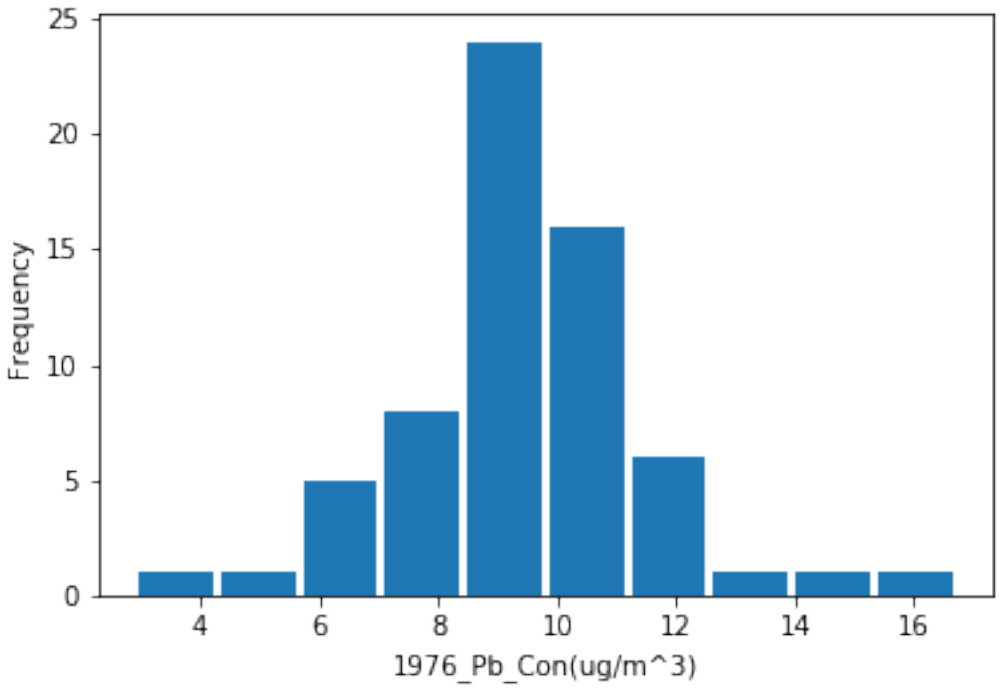
[19]:

| | 1976_Pb_Con(ug/m^3) | 1977_Pb_Con(ug/m^3) |
|---|---|---|
| 0 | 6.7 | 9.5 |
| 1 | 5.4 | 10.7 |
| 2 | 5.2 | 8.3 |
| 3 | 6.0 | 9.8 |
| 4 | 8.7 | 9.1 |

### 7.1 Part a

Construct a frequency distribution and display the results in the form of a histogram. Is this distribution symmetric?

[20]:
```
Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].values
Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].plot(kind='hist',rwidth=0.
 ↪9,bins=10)
plt.xlabel(Problem6_2_10data.columns.values[0])
plt.show()
```




The above histogram appears to be symmetric and normal.

### 7.2 Part b

Calculate the sample mean and sample standard deviation

[21]:
```
dataMean = np.mean(Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].values).
 ↪round(1)
print("The mean of the data is",dataMean)

dataStd = np.std(Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].values).round(1)
print("The standard deviation is",dataStd)
```

```
The mean of the data is 9.4
The standard deviation is 2.1
```

### 7.3 Part c

Locate x_bar and x_bar +/- std dev. How many observations lie within one standard deviation of the mean? How many lie within two?

[22]:
```
std_dev1_count=0
std_dev2_count =0
for a in Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].values:
    if(a<=dataMean+dataStd and a>=dataMean-dataStd):
        std_dev1_count += 1
    if(a<=dataMean+2*dataStd and a>=dataMean-2*dataStd):
```

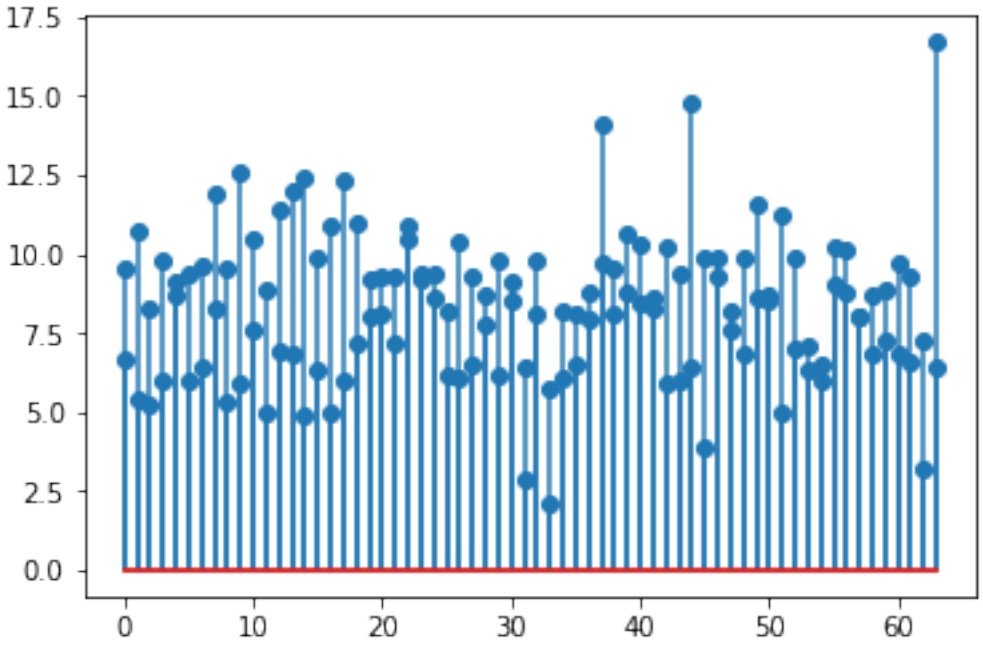
```
        std_dev2_count += 1
print("There are",std_dev1_count,"elements within one standard deviation\n"
      "There are",std_dev2_count,"elements within two standard deviations")
```

```
There are 48 elements within one standard deviation
There are 60 elements within two standard deviations
```

### 7.4 Part d

Using the 1976 and 1977 data create a stem and leaf plot

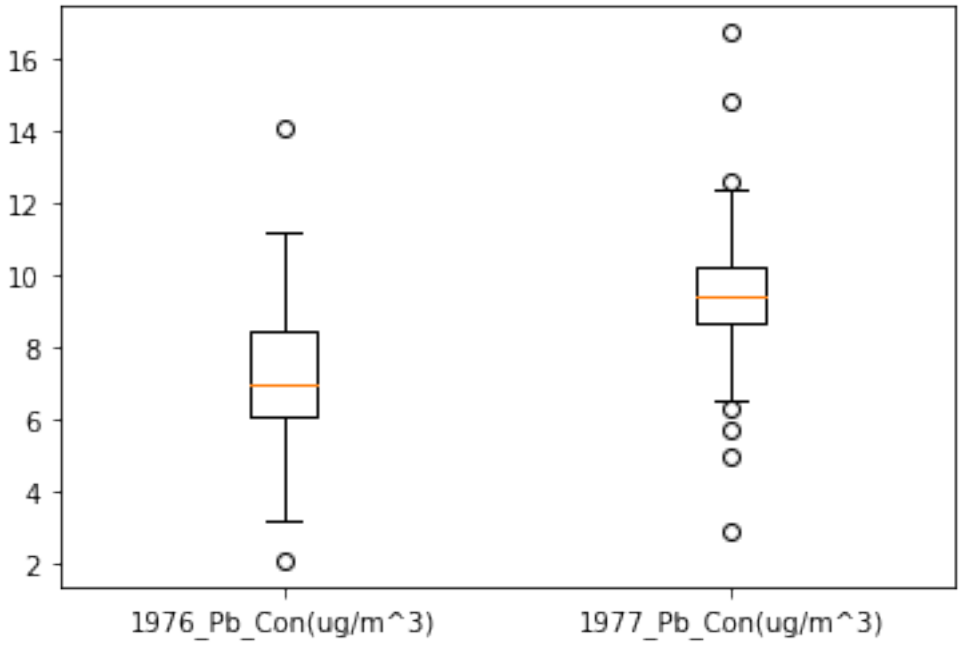
[23]:
```
plt_1976 = plt.stem(Problem6_2_10data.loc[:,"1976_Pb_Con(ug/m^3)"].
 ↪values,use_line_collection=True)
plt_1977 = plt.stem(Problem6_2_10data.loc[:,"1977_Pb_Con(ug/m^3)"].
 ↪values,use_line_collection=True)

plt.show(plt_1976)
plt.show(plt_1977)
```




### 7.5 Part e

Construct box-and-whisker displays of both data sets

[24]:
```
plt.boxplot(Problem6_2_10data.values,labels=Problem6_2_10data.columns)
plt.show()
```




### 7.6 Part f

It appears that the 1977 lead concentration is typically higher than measured in 1976. This can be seen by the boxplot above.